UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Júlia Caroline Arduini de Oliveira

Refatoração Estrutural da Base de Dados do Módulo Gestão RH do sistema SIAF-UFU

Uberlândia, Brasil

2017

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Júlia Caroline Arduini de Oliveira

Refatoração Estrutural da Base de Dados do Módulo Gestão RH do sistema SIAF-UFU

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção do título de Bacharel em Ciência da Computação.

Orientador: Maria Adriana Vidigal de Lima

Universidade Federal de Uberlândia – UFU

Faculdade de Ciência da Computação

Bacharelado em Ciência da Computação

Uberlândia, Brasil

2017

Júlia Caroline Arduini de Oliveira

Refatoração Estrutural da Base de Dados do Módulo Gestão RH do sistema SIAF-UFU

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção do título de Bacharel em Ciência da Computação.

Trabalho aprovado. Uberlândia, Brasil, 12 de dezembro de 2017:

Profa. Dra. Maria Adriana Vidigal de Lima
Orientador

Prof. Dr. Paulo Henrique Ribeiro Gabriel

Prof. Dr. Ronaldo Castro de Oliveira

Uberlândia, Brasil
2017

Agradecimentos

Agradeço à minha adorável orientadora, professora Maria Adriana, pela paciência e sensibilidade de sempre.

Aos meus pais Francisco e Ilze pelo suporte afetivo durante todos os anos da graduação. E ao meu irmão Rodrigo, que tanto amo. Sou grata também aos meus tios Eudes e Pedro, pelos conselhos, confidências e incentivo aos estudos.

Agradeço ao meu supervisor Marcos Alexandre, pelo apoio no desenvolvimento deste trabalho. E também aos demais colegas do CTI.

Agradeço ao meu amor Rafael Martins, pelas palavas gentis e reconfortantes nos momentos mais difíceis.

Meu "muito obrigado" ao meu grande amigo Gustavo Silva, que colaborou para que os anos da faculdade se tornassem inesquecíveis. E a todos os amigos que fizeram parte da minha trajetória na Universidade.

E a todas as pessoas que colaboraram de alguma forma para esta conquista.

"If you plan on being anything less than you are capable of being, you will probably be unhappy all the days of your life."
(Abraham Maslow)

Resumo

Sistemas legados sustentam rotinas vitais de negócio, mas é preciso realizar manutenções periódicas para que eles se mantenham ativos críticos de negócio. Entretanto, estes sistemas podem ser sustentados por banco de dados legados e a modelagem inadequada dos esquemas de relação da base de dados pode gerar problemas no sistema: falta de semântica, inconsistência de dados, grande quantidade de valores nulos e até impossibilidade de representar certas informações. Este trabalho apresenta técnicas de normalização de dados no método de refatoração estrutural que minimizam estes problemas. O processo completo de refatoração de dados é extremamente complexo e requer a dedicação de uma equipe de analistas na organização. Em virtude disto, as regras de normalização de dados aqui apresentadas foram aplicadas na tabela PESSOATIPORH, do subsistema da base de dados do SIAF - Gestão RH - e os resultados obtidos foram generalizados para as demais tabelas do banco de dados do SIAF. Obteve-se ao final do desenvolvimento a semântica clara no esquema, a redução de valores nulos e de valores redundantes, bem como a possibilidade de representar certas informações.

Palavras-chave: refatoração estrutural; normalização; reengenharia de dados; projeto de banco de dados; banco de dados legado.

Lista de ilustrações

Lista de abreviaturas e siglas

1FN	Primeira Forma Normal
2FN	Segunda Forma Normal
3FN	Terceira Forma Normal
3VL	*Three-Valued Logic*
BPMN	*Business Process Modeling Notation*
CFA	Conselho Federal de Administração
COFEN	Conselho Federal de Enfermagem
CPF	Cadastro de Pessoa Física
DDL	*Data Definition Language*
FAEPU	Fundação de Assistência, Estudo e Pesquisa de Uberlândia
GDHS	Gestão de Desenvolvimento Humano em Saúde
HCU	Hospital de Clínicas de Uberlâdia
IBM	*International Business Machines*
PASEP	Programa de Formação do Patrimônio do Servidor Público
PIS	Programa de Integração Social
SIAF	Sistema Administrativo e Financeiro
SQL	*Structured Query Language*
SQL/DS	*Structured Query Language/Data System*
SUS	Sistema Único de Saúde
UFU	Universidade Federal de Uberlândia

Sumário

1 Introdução

1.1 Contextualização

A manutenção de um sistema demanda cerca de 60% dos recursos organizacionais de uma empresa e este percentual é ainda maior quando esta técnica é aplicada a um sistema legado. Estes sistemas, que foram criados há mais de uma década, tornam qualquer alteração no código mais trabalhosa, pois a maioria deles possui baixa qualidade de *software*, além de documentação pobre ou inexistente (PFLEEGER, 2004).

O Sistema Administrativo e Financeiro (SIAF) é um exemplo de sistema legado. Este sistema surgiu da necessidade de tornar públicas as escalas de trabalho dos profissionais do Hospital de Clínicas de Uberlândia - Universidade Federal de Uberlândia (HCU-UFU). O hospital deu início às suas atividades na década de 1970. Na época, as mais de 200 variações de escala eram feitas à mão. Nos dias de hoje o hospital é referência em atendimento de média e alta complexidade no Triângulo Mineiro e região (UFU, 2017b).

Atualmente o SIAF é responsável por toda a parte administrativa e financeira do HCU. O sistema divide-se nas áreas: Materiais, Finanças e Pessoas. A primeira é responsável por gerenciar os materiais de consignação, patrimônios e estoques. A área de Finanças lida com a parte de contabilidade e centros de resultados. A área de Pessoas cuida da parte da gestão de recursos humanos.

Um dos sistemas da área de Pessoas é o Gestão RH, que gerencia o cadastro e os contratos das pessoas vinculadas ao hospital. Destacam-se neste subsistema as tabelas PESSOARH e PESSOATIPORH. A primeira armazena os dados pessoais dos empregados, como documentos pessoais e informações de contato. A tabela PESSOATIPORH pode armazenar 14 tipos diferentes de vínculo que o empregado pode assumir no HCU.

Entretanto, a dificuldade de explicar o significado que os atributos agregam à tabela, como no caso da PESSOATIPORH, pode ser um indício de que esta relação não está bem projetada (ELMASRI; NAVATHE, 2011; AMBLER; SADALAGE, 2006). Além disso, na consulta aos registros armazenados no banco de dados do SIAF, observou-se uma vasta quantidade de valores nulos nas tuplas desta tabela.

No entanto, o que estes nulos representam nas tuplas do banco de dados? O nulo pode assumir os valores “desconhecido”, “não disponível” e “não aplicável”. Quando há um nulo do tipo “desconhecido”, significa que não se tem conhecimento desta informação. O nulo da categoria “não disponível” representa uma informação intencionalmente ocultada.

Já o nulo "não aplicável" é visto como um valor inadequado, uma vez que não existe valor válido para aquele campo (ELMASRI; NAVATHE, 2011).

Os nulos do tipo "não aplicável" podem ser tratados com a remodelagem do esquema de relação (ELMASRI; NAVATHE, 2011). Desta forma, o presente trabalho tem como finalidade a redução dos valores nulos identificados nas tuplas aplicando as técnicas da refatoração estrutural.

Esta remodelagem de dados, feita no processo de manutenção do sistema, é conhecida na engenharia de *software* como reengenharia de dados. Este procedimento que tem a finalidade de corrigir erros de dados, remover registros duplicados e reduzir valores nulos, por sua vez, constitui uma das fases do processo de reengenharia de *software* (SOMMERVILLE, 2011).

A reengenharia de *software* é um procedimento complexo que resulta em um sistema revitalizado a partir do sistema de base, mas sem modificar sua funcionalidade original. Este processo é dividido nas fases de redocumentação, reestruturação, engenharia reversa e modularização do programa (PFLEEGER, 2004).

Percebeu-se que grande parte da complexidade do código do SIAF se deve à necessidade de implementar a semântica das entidades do banco de dados no código do programa. Esta dificuldade poderia ser eliminada com a remodelagem do banco de dados, originando esquemas de relação coesos, com a semântica das relações clara. Por isso, este trabalho propõe a reengenharia de dados antes de remodelar o sistema, pois acredita-se que desta forma a origem do problema é solucionada.

1.2 Motivação

O sistema SIAF é um sistema legado que foi desenvolvido em função de necessidades imediatas dos diversos setores que compõem o HCU ao longo do tempo. Em meio a este cenário, padrões de projeto e documentações foram colocados em segundo plano. Porém, devido à crescente demanda de novas aplicações e, consequentemente, ao aumento da necessidade de operações de manutenção, tornou-se cada vez mais difícil implantar as boas práticas de programação nas funcionalidades que já estavam prontas e em funcionamento.

Em virtude da modelagem atual do banco de dados do SIAF, é exigido que o código do programa implemente não só as regras de negócio do sistema, mas que codifique também a inteligência que captura a semântica das entidades. A semântica das tabelas não pode ser extraída do banco de dados, pois ela não é clara no esquema de relações. Este trabalho extra faz com que qualquer inclusão ou alteração de funcionalidade no sistema seja mais dispendiosa do que deveria ser.

Além das entidades não terem a semântica clara, a organização atual das tabelas na base de dados do SIAF faz com que haja redundância das informações armazenadas e uma quantidade significativamente grande de valores nulos nas tuplas. Outra desvantagem é a impossibilidade de representar certas informações no banco de dados, causada por restrições de multiplicidade. Todos estes fatores colaboram para que a evolução do sistema seja confusa e exaustiva.

1.3 Objetivos

O presente trabalho tem como objetivo principal aplicar as técnicas de refatoração estrutural na base de dados do SIAF, notadamente no módulo Gestão RH. Espera-se que após a remodelagem das relações do SIAF seja possível identificar claramente a semântica atribuída a cada relação do sistema.

Este fator irá permitir que a codificação do SIAF se torne mais intuitiva e descomplicada para os desenvolvedores. Além disto, almeja-se que possam ser armazenadas na base de dados informações que a modelagem atual não permite. Não obstante, pretende-se reduzir as informações redundantes e a quantidade de valores nulos nas tuplas do banco de dados do SIAF. Para alcançar a meta proposta, foram definidos os objetivos específicos a seguir:

1. Identificar problemas no sistema legado SIAF e no banco de dados que armazena suas informações.
2. Avaliar opções para resolver os problemas detectados.
3. Escolher uma entidade conveniente do módulo Gestão RH para realizar o estudo de caso.
4. Aplicar o método de normalização de dados na atividade de refatoração estrutural da relação escolhida.
5. Evidenciar as vantagens da remodelagem aplicada ao objeto de estudo.
6. Generalizar resultados obtidos na aplicação da remodelagem para os demais esquemas de relação do banco de dados do SIAF.

1.4 Organização do trabalho

No Capítulo 2 está descrita a Metodologia utilizada neste trabalho. As bases teóricas: Manutenção de Sistema, Banco de Dados, Falta de Informações, Projeto de Banco de Dados e Refatoração em Banco de Dados são encontradas no Capítulo 3. No Capítulo

4 é apresentado o Banco de Dados do SIAF. O Desenvolvimento do trabalho de remodelagem de dados compõe o Capítulo 5. Finalmente, o Capítulo 6 expõe as Conclusões obtidas do trabalho desenvolvido e apresenta perspectivas para Trabalhos Futuros.

2 Metodologia

É descrita a seguir a sequência das atividades realizadas no desenvolvimento deste trabalho de pesquisa bem como as teorias nas quais este processo baseou-se a fim de atingir aos objetivos propostos na Seção 1.3.

2.1 Estudo do Sistema SIAF e identificação de problemas

Os passos descritos na sequência, de caráter exploratório, foram seguidos a fim de conhecer o escopo geral do objeto de estudo e de fundamentar o desenvolvimento deste trabalho.

Estudo do sistema legado SIAF

Nesta fase foi feita uma análise geral do código fonte das rotinas do SIAF. O intuito principal desta atividade era compreender as regras de negócio, levantar os pontos positivos do sistema, bem como suas falhas e problemas decorrentes de diversas atualizações ao longo do tempo.

Escolha de um dos módulos do SIAF

Após notar a extensão deste sistema legado, elegeu-se o módulo Gestão RH, um subsistema do SIAF, como objeto de estudo desta pesquisa. A escolha deste módulo se deu devido à maior familiaridade com suas funcionalidades, além de concentrar entidades não tão específicas da área hospitalar. Desta forma, a leitura deste trabalho e a compreensão do leitor se dá mais facilmente.

Identificação das maiores dificuldades na codificação do Gestão RH

A identificação das maiores dificuldades do desenvolvimento das funcionalidades do sistema foi feita através do levantamento das maiores dificuldades vivenciadas pelos analistas desenvolvedores do SIAF durante a implementação das rotinas do Gestão RH. Também foi feito o levantamento das maiores dificuldades vivenciadas pelos usuários do Gestão RH, do setor de Gestão de Desenvolvimento Humano em Saúde (GDHS), na utilização do sistema.

Estudo do banco de dados SIAF

Neste estudo utilizou-se o *software* Aqua Data (AQUAFOLD, 2017) para perceber a organização das tabelas do sistema, com enfoque nas utilizadas pelo módulo Gestão RH, e realizar consultas no banco de dados do SIAF. O objetivo era compreender como os dados estavam modelados no banco de dados. Além disso, foi possível observar também a distribuição das informações dentre as diversas tabelas.

Detecção de situações que causam ou que possam vir a causar problemas no Gestão RH

Após analisar o Gestão RH, o banco de dados no qual as informações estão armazenadas e os relatos dos analistas sobre as maiores dificuldades encontradas durante o desenvolvimento do sistema, pôde-se perceber que a origem da complicação estava na modelagem dos dados.

Pelo fato da estrutura do banco de dados ser bem mais engessada do que a estrutura do sistema, este fora compelido a se adequar à modelagem de dados criada há anos. Assim, foram encontradas situações na base de dados que poderiam causar problemas de redundância, incoerência ou de semântica nas informações. Além disso, notou-se uma alta taxa de valores nulos armazenados no banco de dados.

2.2 Avaliação de opções para resolução do problema

Nesta etapa iniciou-se o processo dos estudos teóricos acerca dos assuntos de interesse: Manutenção de *Software*, Sistemas de Banco de Dados, Valores Nulos e Refatoração de Banco de Dados. O objetivo aqui era compreender as teorias nas quais o presente trabalho seria fundamentado.

Estudo do gerenciamento de sistemas de *software*

Em relação a este estudo, utilizou-se para o referencial teórico os livros de Peters e Pedrycz (2011), Pfleeger (2004), Pressman (2011) e Sommerville (2011). Os temas estudados foram: sistemas legados e manutenção de sistemas de *software*. No primeiro tema, os tópicos de atualização, evolução e manutenção de sistemas legados tiveram ênfase no estudo, considerando o sistema Gestão RH nesta categoria. Já na manutenção de *software*, destaca-se a técnica de reengenharia.

Estudo de banco de dados

Para guiar a pesquisa teórica acerca de Banco de Dados utilizou-se as obras de Date (2004), Silberschatz, Korth e Sudarshan (2012) e Elmasri e Navathe (2011). Os

tópicos estudados foram evolução dos bancos de dados e banco de dados relacional. O estudo da evolução dos bancos de dados foi essencial para entender um pouco dos processos históricos que a base de dados do SIAF sofreu e perceber como este banco conseguiu se manter ativo ao longo das décadas. No estudo dos bancos de dados relacionais pôde-se descobrir os aspectos de estrutura, integridade e manipulação que definem o modelo relacional.

Estudo dos valores nulos

A falta de informação no banco de dados e a teoria de como os valores nulos são tratados foram compreendidas pelas mesmas obras do estudo anterior. Percebeu-se neste estudo a dificuldade de realizar operações com valores nulos. Descobriu-se que a abordagem 3VL é utilizada para solucionar tais problemas.

Estudo do projeto de banco de dados

Ainda utilizando os livros Date (2004), Silberschatz, Korth e Sudarshan (2012) e principalmente Elmasri e Navathe (2011), foi possível conhecer os procedimentos principais necessários para aplicação das técnicas de reengenharia de dados no desenvolvimento deste trabalho. Dentre estes procedimentos, destacam-se as diretrizes de projeto para esquemas de relação, o conceito de dependência funcional e, finalmente, as técnicas de normalização.

Estudo do processo de refatoração de banco de dados

Os trabalhos de Ambler e Sadalage (2006) e Domingues (2014) foram essenciais para compreesão das atividades envolvidas no processo de refatoração de banco de dados dados. Durante este estudo pode-se reconhecer a necessidade de refatorar o banco de dados do SIAF. Também promoveu a identificação da categoria de refatoração que seria utilizada neste trabalho.

2.3 Aplicação de técnicas de remodelagem de dados

Com base nos conhecimentos obtidos nas atividades anteriores e nos problemas identificados, optou-se por realizar refatoração estrutural na relação PESSOATIPORH. Na reengenharia de *software* esta técnica é chamada de reengenharia de dados. Para executar a refatoração estrutural foram utilizadas as técnicas de normalização, juntamente com as diretrizes que orientam a criação de projeto para esquemas de relação.

Escolha do escopo para desenvolvimento

Levadas em consideração as principais tabelas do módulo Gestão RH, a relação PESSOATIPORH foi escolhida para exemplificar a aplicação das técnicas de remodelagem, pois é uma tabela importante no sistema e possui casos de uso interessantes.

Estudo aprofundado da tabela PESSOATIPORH

A tabela PESSOATIPORH foi estudada desde seus atributos e a semântica que os mesmos representavam na relação, até a forma com que esta tabela se relaciona com outras tabelas através das chaves estrangeiras. Nesta etapa notou-se alguns padrões de valores nulos para cada tipo de vínculo armazenado nesta tabela. Além disso, pôde-se detectar também alguns problemas de semântica no conjunto dos atributos das relações.

Escolha de um dos tipos da tabela PESSOATIPORH

Através da semântica dos atributos da relação PESSOATIPORH, foi possível detectar 14 tipos de vínculos e, consequentemente, 14 possíveis novas relações. Como mais da metade dos registros cadastrados na tabela PESSOATIPORH são do tipo "funcionário", esta (nova) relação foi escolhida para ser estruturalmente refatorada.

Aplicação da refatoração estrutural

Foram aplicadas na tabela FUNCIONARIO (extraída da relação PESSOATIPORH) as regras de normalização. Utilizaram-se as primeira, segunda e terceira formas normais. As diretrizes do projeto de banco de dados surgiram no decorrer do processo de normalização. Além disto, expôs-se o conjunto resultante de atributos contido na relação FUNCIONARIO após cada forma normal empregada.

Análise dos resultados

A análise geral do desenvolvimento realizado foi feita através da comparação entre a entidade PESSOATIPORH antes e depois da refatoração. Foi possível apresentar vantagens obtidas após a refatoração estrutural da tabela PESSOATIPORH em virtude das dificuldades identificadas antes da refatoração. A análise do resultado obtido foi generalizada para as demais entidades do banco de dados do SIAF.

3 Referencial Teórico

3.1 Mantendo o Sistema

Após a entrega de um *software*, é certo que mudanças sejam feitas para que ele se mantenha um ativo crítico de negócio. Estas mudanças podem ser realizadas por diferentes motivações: correção de erros encontradas em alguma operação, para que o *software* se adéque à plataforma de *hardware* e *software* na qual ele opera, na melhoria de desempenho, ou em virtude de outros requisitos não funcionais (SOMMERVILLE, 2011).

As operações de manutenção demandam de 60% a 70% dos recursos de uma organização de *software*. Osborne e Chikofsky (1990 apud PRESSMAN, 2011) apresentaram uma justificativa para estes valores ao declarar que muitos dos *softwares* dos quais dependemos hoje têm de 10 a 15 anos. E apesar de alguns desses sistemas terem usado técnicas de projeto e codificação (daquela época), o cuidado maior naquele tempo era dado ao tamanho do programa e ao espaço de armazenamento.

Por este motivo, sistemas legados que passaram por muitas modificações a fim de se adaptarem às novas necessidades podem tornar o encargo de manutenção mais trabalhoso (PETERS; PEDRYCZ, 2011).

3.1.1 Sistema legado

Os sistemas legados representam programas de computador desenvolvidos em gerações anteriores, com tecnologia ultrapassada, mas que supriram as necessidades do cliente na época (PETERS; PEDRYCZ, 2011). Por serem antigos, alguns deles possuem baixa qualidade de *software*, no sentido de que padrões de projeto modernos não existiam ou não eram bem compreendidos no período de sua implementação (PRESSMAN, 2011).

Apesar de possuir projetos despreparados para expansão, codificação altamente acoplada e documentação vaga ou inexistente, sistemas legados sustentam rotinas vitais de negócio (PRESSMAN, 2011).

3.1.2 Manutenção de *software*

Independentemente do tempo de vida, todo sistema de *software* deve passar por manutenções contínuas para permanecer útil. Manutenção de *software* é todo procedimento de modificação do sistema realizado após sua entrega aos usuários finais (SOMMERVILLE, 2011).

As organizações têm investido cada vez mais em manutenção de *software*. Pfleeger (2004) mostra um pouco desse histórico de ascensão no investimento:

> Na década de 70, a maior parte do orçamento de um sistema de *software* era gasta no desenvolvimento. A proporção entre o valor investido no desenvolvimento e o investido na manutenção foi revertida, nos anos 80, e várias estimativas supõem que a manutenção representa de 40 a 60 por cento do custo total do ciclo de vida de um sistema. [...] As estimativas atuais sugerem que, no ano 2000, os custos de manutenção aumentaram em até 80 por cento do custo referente ao tempo de vida de um sistema.

Realizar a manutenção de um *software*, de forma similar às atividades de desenvolvimento, requer análise de requisitos, avaliação do sistema e do projeto do programa, implementação e revisão do código, além de testes das alterações e atualização da documentação (PFLEEGER, 2004). Segundo Peters e Pedrycz (2011), os três tipos de manutenção que levam um *software* a evoluir são:

- Manuteção corretiva - tem como finalidade realizar correções de problemas no código proveniente de falhas e erros descobertas no *software*.
- Manutenção adaptativa - adaptar o *software* em relação ao ambiente e às tecnologias no qual ele será implantado.
- Manutenção perfectiva - abrange inserção, modificação e extensão do *software* para implementar novos requisitos de negócio e atender às necessidades do cliente.

Lientz e Swanson (1980 apud PFLEEGER, 2004) pesquisaram como gerentes em 487 organizações diferentes lidam com as atividades de manutenção de um sistema e descobriram que a maior parte dos esforços são provenientes da manutenção perfectiva. Encontra-se na Figura 1 a distribuição dos esforços para as manutenções corretiva, adaptativa e perfectiva.

Técnicas de Manutenção

São encontradas maiores dificuldades nas atividades de manutenção quando se trata de um sistema legado mais antigo, onde a inteligibilidade do programa pode ter sido afetada pelas várias alterações de melhoria de desempenho e uso de espaço, ou deterioração da estrutura inicial do programa ao longo dos anos. Com a reengenharia de *software* é possível tornar sistemas legados de *software* mais fáceis de serem mantidos implantando melhorias na estrutura e aumentando a inteligibilidade (SOMMERVILLE, 2011).

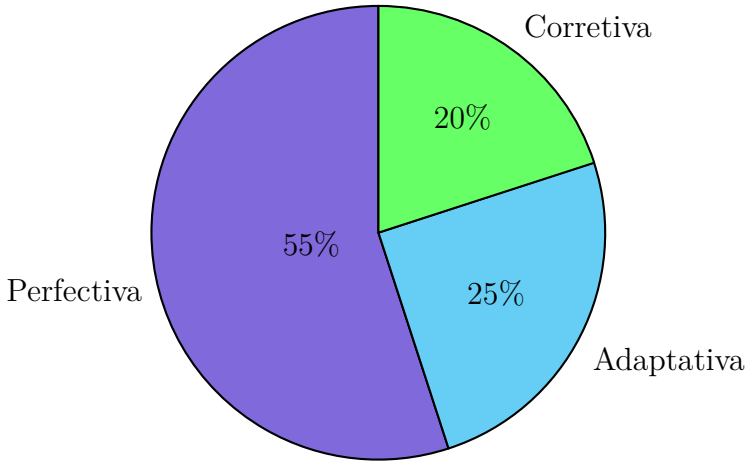

Figura 1 – Distribuição do esforço de manutenção, adaptado de Peters e Pedrycz (2011).

A reengenharia de *software* é uma das etapas do processo de rejuvenescimento do *software* que, de acordo com Pfleeger (2004), assume o propósito de aumentar a qualidade de todo o sistema. Segue a lista completa de todas as etapas deste processo.

- Redocumentação
- Restruturação
- Engenharia Reversa
- Reengenharia

As técnicas de redocumentar e reestruturar o sistema dependem basicamente do código-fonte. A redocumentação do sistema apenas extrai informações do código-fonte para auxiliar em sua manutenção. Já na reestruturação, converte-se código mal estruturado para bem estruturado (PFLEEGER, 2004). A ideia da engenharia reversa é extrair a documentação completa do projeto e informações do código-fonte a partir de um projeto mal estruturado (PRESSMAN, 2011). O processo de reengenharia de *software* é o mais abrangente das abordagens de evolução de sistema, envolvendo análise e compreensão do sistema, sua reconstrução e reimplementação (PETERS; PEDRYCZ, 2011).

Redocumentação

A redocumentação do sistema implica na análise do código-fonte para extrair informações que serão utilizadas na elaboração fiel da documentação, isto é, o registro do que o sistema de fato faz (PFLEEGER, 2004).

Reestruturação

A fase de reestruturação altera o código a fim de facilitar no entendimento e na modificação do *software*. A representação interna - extraída do código-fonte não estruturado - é simplificada através das regras de transformação e assim obtém-se o código-fonte estruturado (PFLEEGER, 2004).

Engenharia reversa

Similar à redocumentação, partindo do código-fonte, a engenharia reversa produz informações da especificação e do projeto do sistema (PFLEEGER, 2004). É o processo que procura fornecer uma representação de programa de nível mais alto de abstração que o nível do código-fonte. O aumento do nível de abstração da representação facilita o entendimento do programa (PRESSMAN, 2011).

É importante realizar a reestruturação do código antes do procedimento de engenharia reversa para que a extração das abstrações do processamento, da interface e da base de dados não sejam poluídas com informações redundantes ou incoerentes. A última parte deste processo se resume em refinar e simplificar a especificação inicial, obtida na etapa anterior, de extração de abstrações. Obtém-se então a especificação final significativa do processamento executado, da interface de usuário e das estruturas de dados do programa (PRESSMAN, 2011).

Reengenharia

O processo de reengenharia de *software* é o único que de fato resulta em um *software* "novo". Novo no sentido de gerar um código-fonte revitalizado, mas sem comprometer a função original do sistema (PFLEEGER, 2004). Para este fim, o sistema de *software* é examinado e alterado de modo a ser reconstruído e reimplementado para originar um novo formato. Isto é, a documentação é refinada de forma a esclarecer o que o sistema faz, elabora-se e reestrutura o projeto do sistema para atingir uma forma mais razoável e então o *software* é reestruturado (PETERS; PEDRYCZ, 2011).

Sommerville (2011) levanta duas vantagens na reengenharia em relação à substituição do sistema:

1. Risco reduzido: A criação de um novo *software* crítico de negócio pode ser muito arriscada, pois podem ocorrer erros na especificação ou no desenvolvimento do sistema. Além disto, atrasos podem causar a perda do negócio.

2. Custo reduzido: Apesar de dispendiosa, o gasto em recursos para a reengenharia pode ser substancialmente inferior aos gastos de desenvolver um *software* do zero.

Os procedimentos de reengenharia de *software* se dividem em cinco etapas (algumas delas já foram introduzidas anteriormente): tradução do código-fonte, engenharia reversa, reestruturação, modularização do programa e reengenharia de dados (SOMMERVILLE, 2011).

A primeira parte, tradução de código-fonte, consiste em traduzir o código-fonte da linguagem de código antiga para a versão mais nova, ou para outra linguagem (SOMMERVILLE, 2011). Esta etapa é bastante útil especialmente se o sistema em questão, passando pelo processo de reengenharia, for um sistema legado. É bem provável que a linguagem do código-fonte do sistema tenha caído em desuso ou impõe muitas limitações em relação às novas tecnologias com a qual ela opera. Então a fase de tradução de código-fonte tem a oportunidade de implantar uma linguagem mais moderna e robusta.

Na engenharia reversa, as informações são extraídas a partir do código a fim de auxiliar na documentação das funcionalidades. Paralelamente ao processo de engenharia reversa ocorre a reestruturação do código com o objetivo de aumentar a legibilidade do código. A modularização de programa cruza a documentação obtida no processo de engenharia reversa com o programa reestruturado, oriundo da reestruturação. Esta etapa agrupa partes relacionadas do programa e remove redundâncias quando necessário (SOMMERVILLE, 2011).

Um programa reconstruído, originado da etapa de modularização de programa, demanda a etapa final da reengenharia de *software*, a reengenharia de dados. Neste estágio, os dados são modificados para se adaptarem ao novo programa, ou seja, redefinição dos esquemas de banco de dados. Isto pode implicar na correção de erros dos dados, remoção de registros duplicados, entre outras alterações (SOMMERVILLE, 2011).

3.2 Banco de Dados

Um sistema de banco de dados é um sistema de computador responsável pelo armazenamento de registros de modo que estas informações possam ser consultadas e alteradas pelos usuários. Estes registros servem como auxílio em processos de negócios de indivíduos ou organizações (DATE, 2004).

3.2.1 Evolução dos bancos de dados

Herman Hollerith, no início do Século XX, inventou os **cartões perfurados** a fim de registrar dados de censo nos Estados Unidos. Para processamento dos cartões e tabulação dos resultados utilizou-se sistemas mecânicos e logo os cartões se tornaram tendência de recurso para inserção de dados em computador (SILBERSCHATZ; KORTH; SUDARSHAN, 2012).

A evolução do armazenamento e processamento de dados ao longo do tempo, segundo Silberschatz, Korth e Sudarshan (2012) são apresentadas na sequência.

- Década de 1950 e início da década de 1960: as **fitas magnéticas** foram os dispositivos de armazenamento de dados que marcaram este período. Uma nova fita era escrita decorrente dos dados de uma ou mais fitas lidas durante o processamento. Também era possível ler cartões perfurados em sincronia com fitas magnéticas. Utilizava-se deste recurso, por exemplo, para realizar alterações nos dados de uma fita: as modificações eram inseridas nos cartões perfurados e após leitura simultânea com as fitas, as informações atualizadas eram escritas na recém-criada fita magnética. Além do modo de leitura ser apenas sequencial, a memória principal era bem inferior ao tamanho dos dados, o que forçava uma ordem específica dos dados no processamento.

- Final da década de 1960 e década de 1970: o acesso direto aos dados no processamento tornou-se possível graças ao surgimento dos **discos rígidos** no final da década de 1960. A obrigatoriedade dos dados em sequência teve fim, uma vez que agora os dados poderiam ser acessados em qualquer posição do disco em dezenas de milissegundos. Estruturas de dados como listas e árvores criadas em bancos de dados em rede e hierárquicos tornaram-se possíveis de armazenar no disco. Neste período destacou-se a definição de modelo relacional e métodos procedurais de consultar seus dados que originou os bancos de dados relacionais.

- Década de 1980: apesar da ideia de ocultar os detalhes de implementação do desenvolvedor ser bastante interessante, o modelo relacional não teve aceitação imediata por apresentar menor desempenho em relação aos bancos de dados predecessores (de rede e hierárquicos). Até que um projeto da IBM (*International Business Machines*) *Research*, o *System R*, lançou o primeiro sistema gerenciador de banco de dados relacional da IBM, o SQL/DS (*Structured Query Language/ Data System*). Logo lançaram também os primeiros sistemas de banco de dados relacionais comerciais que teriam papel essencial para o desenvolvimento de técnicas para processamento eficiente de consultas declarativas: IBM DB2, Oracle, Ingrees e DEC Rdb. Devido à melhoria no desempenho e à facilidade de usar, os bancos de dados relacionais tomaram lugar dos bancos de dados de rede e hierárquicos. Com os bancos de dados relacionais os programadores podiam atuar sobre o nível lógico sem a necessidade de se preocuparem com tarefas de baixo nível, uma vez que elas eram feitas automaticamente pelo banco de dados. Em virtude destas vantagens, o modelo de **banco de dados relacional** se tornou, e permanece até hoje, como o modelo mais difundido. Iniciou-se nesta década as pesquisas sobre banco de dados paralelos, distribuídos e orientados a objeto.

- Início da década de 1990: Uma nova área de aplicação para banco de dados surgiu da necessidade de criação de consulta e suporte a decisão, proporcionada pela **linguagem SQL**. Também iniciou-se a incorporação de suporte relacional de objeto aos bancos de dados.

- Final da década de 1990: As taxas de processamento de transação, confiabilidade e disponibilidade dos bancos de dados tiveram que aumentar abruptamente para suportar o crescimento da ***World Wild Web***. Além disto, também era esperado que os bancos de dados aceitassem interfaces da *Web* para dados.

- Década de 2000: Surgiram as **linguagens XML e XQuery**. Também houve expansão da "computação autônoma" a fim de diminuir o empenho investido na administração do sistema. Sistemas *open-source* de banco de dados PostgreSQL e MySQL também passaram a ser mais utilizados. No final da década notou-se o crescimento no uso de banco de dados especializados para análise de dados. A última tendência foram as técnicas de mineração de dados para recomendação de produtos e exibição automática de anúncios de interesse do usuário em páginas *Web*.

3.2.2 Banco de dados relacional

Os bancos de dados relacionais são apoiados pelo **modelo relacional** de dados, um sistema que define essencialmente aspectos estrutural, de integridade e manipulativo. O aspecto estrutural diz que os dados no banco de dados são percebidos pelo usuário como tabelas. O segundo elemento garante que estas tabelas satisfazem a determinadas restrições de integridade. O fator manipulativo disponibiliza operadores para que manipulações nessas tabelas sejam realizadas (DATE, 2004).

Tabelas

As **tabelas** que definem o fator estrutural do modelo relacional podem tanto representar as informações dos dados, quanto as **relações** estabelecidas entre elas. Cada tabela possui diversas colunas, também conhecidas como **atributos**, cujo nome é único. O conjunto das colunas de uma tabela descreve o tipo específico de **registro**, ou tupla, daquela tabela. O tipo de registro, por sua vez, estabelece a quantidade fixa de campos da tabela (SILBERSCHATZ; KORTH; SUDARSHAN, 2012).

Por exemplo, uma empresa pode armazenar dados pessoais de seus empregados em uma tabela PESSOA que terá dados como CPF (Cadastro de Pessoa Física), nome, idade e escolaridade. Deve haver também uma tabela CARGO para armazenar os cargos da empresa e seus respectivos salários. Os esquemas para estas relações são, respectivamente, PESSOA(CPF, nome, idade, escolaridade) e CARGO(cargo, salário) (SILBERSCHATZ; KORTH; SUDARSHAN, 2012).

É possível também associar uma instância de PESSOA a uma ou mais instâncias de CARGO. Assim, uma tabela VÍNCULO com os atributos CPF (do esquema PESSOA), cargo (da tabela CARGO), data_admissão e data_desligamento expressa os vínculos empregatícios de cada contratação. Seu esquema fica: VÍNCULO(CPF, cargo, data_admissão, data_desligamento) (SILBERSCHATZ; KORTH; SUDARSHAN, 2012).

A Figura 2 a seguir mostra as tabelas PESSOA, CARGO e VÍNCULO, que foram definidas anteriormente, com alguns dados de exemplificação. Nota-se que algumas linhas do atributo data_desligamento do esquema VÍNCULO estão preenchidas com o valor "*null*". O **valor nulo** representa um valor desconhecido ou inexistente (SILBERSCHATZ; KORTH; SUDARSHAN, 2012).

PESSOA

CPF	nome	idade	escolaridade
111.111.111-11	Maria	34	Mestrado
222.222.222-22	Adriana	21	Ensino Fundamental
333.333.333-33	Marcos	45	Doutorado

CARGO

cargo	salário
Funcionário	R$ 2000,00
Professor	R$ 5000,00
Estagiário	R$ 600,00

VÍNCULO

vinc#	CPF	cargo	data_admissão	data_desligamento
V1	111.111.111-11	Funcionário	07/01/2013	*null*
V2	222.222.222-22	Estagiário	06/02/2017	*null*
V3	333.333.333-33	Funcionário	03/07/2000	31/12/2012
V4	333.333.333-33	Professor	15/02/2012	*null*

Figura 2 – Banco de dados de pessoas e seus vínculos (amostra de valores).

Neste caso (e em muitos outros) não se sabe qual é o tipo de nulo em cada tupla, uma vez que o campo data_desligamento pode estar nulo porque este valor não foi atualizado quando o empregado fora desligado, ou porque de fato esta data não existe pois o empregado ainda não foi desligado. Tratamento de nulos em banco de dados é um assunto complexo e será abordado com mais detalhes na próxima seção (Seção 3.3).

Chaves

O conceito de chave no banco de dados relacional é muito importante, pois são como a digital de uma tabela e é a forma com a qual as relações são referenciadas. As chaves são atributos que identificam cada tupla da relação como única. Isto faz com que

nenhuma tupla seja idêntica a outra. Uma **superchave** é um subconjunto dos atributos de uma relação no qual os valores desse conjunto de atributos não se repetem (SILBERSCHATZ; KORTH; SUDARSHAN, 2012). O atributo CPF da tabela PESSOA, por exemplo, é uma superchave pois identifica unicamente cada instância da relação PESSOA.

Quando há mais de uma superchave em uma relação, elas são denominadas **chaves candidatas**. Então o projetista escolhe a **chave primária** a partir das chaves candidatas para representar o identificador único daquela relação. É importante escolher uma chave primária onde seus valores nunca (ou raramente) mudem (SILBERSCHATZ; KORTH; SUDARSHAN, 2012). As chaves candidatas que não forem escolhidas pelo projetista são chamadas de **chaves únicas** (ELMASRI; NAVATHE, 2011).

Restrições

As **restrições de integridade** foram criadas para que alterações prejudiciais à consistência dos dados no banco de dados fossem evitadas (SILBERSCHATZ; KORTH; SUDARSHAN, 2012). Na sequência serão apresentadas restrições baseadas em esquema (ou restrições explícitas), que podem ser especificadas na DDL (do inglês *Data Definition Language*: Linguagem de Definição de Dados) (ELMASRI; NAVATHE, 2011).

Restrições de domínio, também chamadas de restrições de atributo, estabelecem o tipo específico de dado (INTEGER, FLOAT, CHARACTER, etc.) que cada atributo do registro aceita (DATE, 2004). Na base de dados da Figura 2, a tabela CARGO é mapeada com os tipos cargo: CHARACTER e salário: DOUBLE.

Além das chaves primárias identificarem unicamente as entidades, nenhum de seus campos podem assumir valor nulo, definindo assim a **restrição de integridade de entidade**. Isto ocorre porque não seria possível discernir entre duas (ou mais) tuplas com valores nulos na chave primária, pois as operações com valores nulos é singular (mostrado na Seção 3.3.2) (ELMASRI; NAVATHE, 2011).

Existem relações que incluem em seus atributos chaves primárias de outras relações. A tabela VÍNCULO, por exemplo, possui os atributos CPF e cargo, ambos chaves primárias de outras relações (PESSOA e CARGO). Estes atributos são denominados **chaves estrangeiras** da relação VÍNCULO (SILBERSCHATZ; KORTH; SUDARSHAN, 2012).

Uma **restrição de integridade referencial** da relação VÍNCULO para as relações PESSOA e CARGO impõe a existência de pelo menos uma instância das relações referenciadas (PESSOA e CARGO) que contenham os valores das chaves estrangeiras da relação referenciadora (VÍNCULO) (SILBERSCHATZ; KORTH; SUDARSHAN, 2012). Se as instâncias de PESSOA e/ou CARGO cujos valores referenciados em VÍNCULO ainda não foram criadas, então será atribuído o valor *null* a estes atributos (ELMASRI;

NAVATHE, 2011).

Restrições de integridade semântica identificam restrições às variáveis de relação. Isto é, determina os valores válidos para uma variável de relação. Um exemplo na base de dados da Figura 2 é "cargo(Estagiário) deve ter escolaridade(Ensino Fundamental)" (DATE, 2004).

Restrição de dependência funcional indica que existe uma relação funcional entre dois conjuntos de atributos A e B. Ou seja, o conjunto de valores de A especifica um conjunto único de valores de B em qualquer ocorrência no banco (ELMASRI; NAVATHE, 2011).

Restrições de transição são restrições dinâmicas aplicadas nas transições (qualquer alteração no banco de dados) com a finalidade de manter o banco de dados em um estado válido, íntegro (ELMASRI; NAVATHE, 2011). Para exemplo no banco de dados da Figura 2, tem-se "Escolaridade em PESSOA não deve decrescer". Isto indica, por exemplo, que a atualização escolaridade(Doutorado) para escolaridade(Ensino Fundamental) não é permitida.

Operações

Os **operadores** mais importantes são de restrição, projeção e junção (DATE, 2004). A operação de restrição, também conhecida como seleção, seleciona linhas de uma tabela. A apresentação de colunas é feita pela operação de projeção. E a junção utiliza um valor comum entre duas tabelas para unificá-las. O resultado dessas operações no banco de dados de pessoas e seus vínculos da Figura 2 são apresentadas em seguida, nas Figuras 3, 4 e 5.

Operação de restrição: PESSOAs nas quais idade $>$ 30

Resultado:

CPF	nome	idade	escolaridade
111.111.111-11	Maria	34	Mestrado
333.333.333-33	Marcos	45	Doutorado

Figura 3 – Operação de restrição na base de dados de pessoas e seus vínculos.

Operação de projeção: PESSOAs sobre CPF, nome

Resultado:

CPF	nome
111.111.111-11	Maria
222.222.222-22	Adriana
333.333.333-33	Marcos

Figura 4 – Operação de projeção na base de dados de pessoas e seus vínculo.

Operação de junção: VÍNCULOs e CARGOs sobre cargo
Resultado:

vinc#	CPF	cargo	data_admissão	data_desligamento	salário
V1	111.111.111-11	Funcionário	07/01/2013	*null*	R$ 2000,00
V2	222.222.222-22	Estagiário	06/02/2017	*null*	R$ 600,00
V3	333.333.333-33	Funcionário	03/07/2000	31/12/2012	R$ 2000,00
V4	333.333.333-33	Professor	15/02/2012	*null*	R$ 5000,00

Figura 5 – Operação de junção na base de dados de pessoas e seus vínculo.

3.3 Falta de Informações

É necessário abordar, de alguma forma, a falta de informação que naturalmente existe entre as comunicações (DATE, 2004). Um valor *null* no banco de dados pode assumir três diferentes significados: "desconhecido", "não disponível" ou "não aplicável". O nulo recebe um valor "desconhecido" quando o atributo existe, mas não se sabe seu valor. O nulo que representa valor "não disponível" também existe, mas é intencionalmente ocultado. Já o nulo que denota valor "não aplicável" é interpretado como inadequado, uma vez que aquele atributo não deveria pertencer àquela tupla (ELMASRI; NAVATHE, 2011).

3.3.1 Problemas relacionados a valores nulos

Os valores nulos apresentam problemas particulares em operações aritméticas e de comparações. Seja operação de adição, subtração, multiplicação ou divisão, se pelo menos um dos valores for nulo o resultado também o será. Encontra-se dificuldade também em obter a saída de comparações entre valores quando há valor nulo envolvido. Afirmar que a sentença "$7 < null$" é verdadeiro seria tão incorreto quando dizer que ela é falsa. Afinal, não se sabe o valor representado por este *null*(SILBERSCHATZ; KORTH; SUDARSHAN, 2012).

Como seria então o resultado de uma consulta utilizando a cláusula *where*, por exemplo, caso encontrasse uma sentença do tipo "7 < *null*"? A SQL soluciona estes casos de valores nulos em operações aritméticas e operações de comparações com a lógica trivalorada (SILBERSCHATZ; KORTH; SUDARSHAN, 2012).

3.3.2 Abordagem 3VL

A lógica trivalorada, também conhecida na forma reduzida como "3VL" (do inglês *Three-Valued Logic*), é frequentemente usada para solucionar problemas de categorizar o resultado de operações relacionais com valores nulos. Esta abordagem possui este nome pois oferece o valor verdade **desconhecido**, também representado por "unk" (abreviação de "desconhecido" em inglês - "*unknown*"), além dos outros dois valores **verdadeiro** e **falso**.

Em resposta à pergunta apresentada ao final da Seção 3.3.1, a cláusula *where* agrega ao resultado apenas as sentenças avaliadas como verdadeira e ignora proposições falsas ou desconhecidas. É possível, entretanto, extrair tuplas com valor *null* (ou diferente de) em atributos especificados pelo usuário (SILBERSCHATZ; KORTH; SUDARSHAN, 2012). As tabelas verdade de 3VL para as operações AND e OR e para as cláusulas NOT e IS NULL são mostradas na Figura 6 (v = verdadeiro, f = falso e u = unk) (DATE, 2004).

AND	v	f	u
v	v	f	u
f	f	f	f
u	u	f	u

OR	v	f	u
v	v	v	v
f	v	f	u
u	v	u	u

NOT	
v	f
f	v
u	u

IS NULL	
v	f
f	f
u	v

Figura 6 – Tabelas verdade 3VL. Adaptado de Date (2004).

É possível fazer uso do predicado ***is null*** para realizar uma consulta no banco de dados a fim de exibir os registros da tabela VÍNCULO que estão sem data de desligamento. A consulta ficaria assim:

```
SELECT *
FROM VINCULO
WHERE data_desligamento is null
```

De forma análoga, está disponível o predicado ***is not null*** com a finalidade de consultar tuplas onde este atributo possui algum valor, ou seja, é diferente de *null*.

3.4 Projeto de Banco de Dados

Como medir a qualidade de um projeto de banco de dados? Com os fundamentos de dependências funcionais é possível identificar se o conjunto dos agrupamentos de atributos no esquema de relação é adequado, ou se é possível melhorá-lo (ELMASRI; NAVATHE, 2011).

As boas práticas de esquemas de relação podem ser aplicadas no **nível lógico**, também conhecido como conceitual, ou no **nível de implementação**, chamado também de armazenamento físico. O nível lógico representa as interpretações dos esquemas de relação e o que os atributos significam para os usuários. O nível de implementação diz respeito ao armazenamento físico dos esquemas das relações de base (ELMASRI; NAVATHE, 2011).

O intuito do projeto de banco de dados relacional é **preservar a informação** e manter a **redundância mínima**. A preservação da informação é mensurada pela manutenção dos tipos de atributo, tipos de entidade e tipos de relacionamento, detectados no projeto conceitual. Minimizar a redundância significa diminuir o armazenamento de informações redundantes. Isto irá consequentemente reduzir a quantidade necessária de atualizações nas duplicatas da mesma informação, a fim de manter a consistência nos dados (ELMASRI; NAVATHE, 2011).

3.4.1 Diretrizes informais de projeto para esquemas de relação

Segundo Elmasri e Navathe (2011), algumas diretrizes informais para medir a qualidade de projeto do esquema da relação são:

1. Garantir que a semântica dos atributos seja clara no esquema;
2. Reduzir a informação redundante nas tuplas;
3. Reduzir os valores *null* nas tuplas;
4. Reprovar a possibilidade de gerar tuplas falsas.

Diretriz 1

A primeira diretriz quer dizer que o conjunto de atributos de um esquema deve ter alguma razão semântica para justificar o agrupamento. Além disto, a facilidade de explicar um esquema de relação é um bom indício de que ele está semanticamente correto. Combinar atributos de diversos tipos de entidade e de relacionamento em apenas uma relação torna o esquema semanticamente ambíguo e causa dificuldades em explicar o que ela representa (ELMASRI; NAVATHE, 2011).

EMPREGADO

CPF	nome	idade	escolaridade	cargo	salário
111.111.111-11	Maria	34	Mestrado	Funcionário	R$ 2000,00
222.222.222-22	Adriana	21	Ensino Fundamental	Estagiário	R$ 600,00
333.333.333-33	Marcos	45	Doutorado	Funcionário	R$ 2000,00
333.333.333-33	Marcos	45	Doutorado	Professor	R$ 5000,00

Figura 7 – Relação EMPREGADO com anomalias de atualização.

A relação EMPREGADO da Figura 7 apresenta a violação da primeira diretriz, pois não é possível afirmar com certeza se esta relação representa as pessoas ou os cargos desta empresa. Como foi mostrado na Figura 2, os dados pessoais de cada pessoa podem ser agrupados em uma relação (PESSOA), bem como as informações pertinentes aos cargos da empresa em outra relação (CARGO).

Diretriz 2

O projeto de esquema tem também o propósito de minimizar o espaço utilizado pelas relações no armazenamento, que está diretamente ligado ao modo como os atributos são agrupados. Na tabela EMPREGADO da Figura 7, os dados relacionados a um CPF são repetidos para cada registro que possui o mesmo valor neste atributo. Já na modelagem da Figura 2, as informações pessoais ligadas ao CPF aparecem uma única vez por valor de CPF, assim como as informações relacionadas ao atributo cargo na tabela CARGO são apresentadas unicamente por valor deste atributo (ELMASRI; NAVATHE, 2011).

Um esquema de relações onde as relações da base são armazenadas em junções naturais, como a relação EMPREGADO da Figura 7, geram problemas de **anomalias de atualização**. Por sua vez, dividem-se em anomalias de: inserção, exclusão e modificação (ELMASRI; NAVATHE, 2011).

Inserir um registro na tabela EMPREGADO pode ser problemático de duas formas. A primeira surge na necessidade de registrar uma pessoa que será empregada na empresa; e a segunda, de registrar um novo cargo.

No primeiro cenário tem-se a inserção de uma pessoa cujo cargo ainda não fora criado. Então as colunas cargo e salário desta relação recebem o valor *null* nas tuplas inseridas. Isto gera confusão na interpretação da relação EMPREGADO, já que ela traz apenas os dados pessoais do empregado e falta informações sobre a sua posição na empresa. Não fica mais simples se o cargo existe, pois é preciso ter atenção na atribuição dos valores de CARGO para não haver incoerência entre os dados inseridos para o mesmo valor cargo.

Na segunda situação, decorrente da necessidade de registrar um novo CARGO, pode ser que este CARGO ainda não tenha nenhuma pessoa que o ocupe. Desta forma, as colunas CPF, nome, idade e escolaridade receberão o valor *null*. Mais uma vez gerando imprecisão na semântica da relação EMPREGADO, com dados de CARGO sem uma PESSOA atribuída a ele.

A anomalia de exclusão se deve ao fato de que a relação EMPREGADO amarra duas entidades. Assim que a última tupla da pessoa que ocupa um determinado cargo for removida da tabela EMPREGADO, as informações deste cargo serão extintas do banco de dados (ELMASRI; NAVATHE, 2011).

Caso haja mudança em algum valor dos atributos de PESSOA ou CARGO, será necessário replicar esta mudança em todos as tuplas onde há ocorrência deste valor. Atualização da escolaridade do EMPREGADO Marcos, por exemplo, resultaria na necessidade de atualização em duas tuplas, já que ele é um dos empregados que possui dois cargos na empresa (ELMASRI; NAVATHE, 2011).

Tais anomalias poderiam ser facilmente evitadas se o projeto de esquemas de relação de base fosse feito sem junções naturais entre entidades cujos significados são diferentes. Retoma-se aqui a importância da primeira diretriz (ELMASRI; NAVATHE, 2011).

Diretriz 3

Valores nulos nas tuplas podem surgir em grande número nas relações mais robustas se muitos dos atributos não se aplicarem a todas as tuplas desta relação. As implicações resultantes são: desperdício de espaço de armazenamento; problemas na definição do significado dos atributos; e problemas com a especificação de operações de junção (ELMASRI; NAVATHE, 2011).

Como mostrado na Seção 3.3, os nulos podem assumir os valores “não se aplica”, “desconhecido” e “ausente”. Desta forma, não é possível discernir segura e prontamente qual tipo de nulo o valor *null* representa na tupla (ELMASRI; NAVATHE, 2011).

Dito isto, é importante evitar ao máximo a inclusão de atributos em relações de base cujos valores serão frequentemente nulos. É essencial para o projetista do esquema a percepção de quando um grupo de atributos devem ser incluído em uma relação e quando deve-se criar outra relação para agrupar este conjunto e referenciá-la através da chave (ELMASRI; NAVATHE, 2011).

Diretriz 4

A geração de tuplas falsas é decorrente da operação de junção entre relações utilizando atributos que não são chaves primárias nem chaves estrangeiras nestas relações. O projeto dos esquemas de relação deve ser feito de forma tal que seja possível unir duas relações sem que tuplas falsas sejam produzidas. Relações com atributos equivalentes que não sejam chave primária ou chave estrangeira, portanto, devem ser evitadas a fim de garantir que não surjam tuplas falsas (ELMASRI; NAVATHE, 2011).

3.4.2 Dependências funcionais

Dependência funcional define uma restrição entre dois grupos de atributos no banco de dados. Assuma um esquema de relação $R = \{A_1, A_2, ..., A_n\}$, onde A_i representa um atributo e n o número total de atributos da relação. A dependência funcional entre os conjuntos de atributos X e Y é representada por $X \rightarrow Y$. Isto significa que dado as tuplas t_i e t_j, onde $i = j$, se as tuplas t_i e t_j possuem o mesmo valor para o conjunto X, $t_i[X] = t_j[X]$, então elas também deverão ter o mesmo valor para o conjunto Y, $t_i[Y] = t_j[Y]$, para qualquer tupla i e j da relação R (ELMASRI; NAVATHE, 2011).

A dependência funcional é uma propriedade da semântica dos atributos. Assim, é preciso ter conhecimento sobre todo o banco de dados para entender como os atributos relacionam entre si para, só então, afirmar com convicção onde estão as dependências funcionais. Ao detectar uma dependência funcional, esta deve ser especificada como uma restrição a fim de melhor especificar um esquema de relação (ELMASRI; NAVATHE, 2011).

Considerando a relação EMPREGADO da Figura 7, as seguintes dependências funcionais podem ser extraídas: $CPF \rightarrow \{nome, idade, escolaridade\}$ e $cargo \rightarrow salário$. A primeira dependência mostra que o valor do CPF do empregado determina exclusivamente o nome, a idade e a escolaridade deste empregado. A segunda dependência funcional especifica que o salário é determinado de maneira funcional por cargo (ELMASRI; NAVATHE, 2011).

Elmasri e Navathe (2011) trazem na relação $R(A, B, C, D)$ da Figura 8 as possíveis dependências funcionais que podem ser extraídas: $B \rightarrow C$; $C \rightarrow B$; $\{A, B\} \rightarrow C$; $\{A, B\} \rightarrow D$; e $\{C, D\} \rightarrow B$ Também é possível notar onde não haverá dependência funcional por apresentar violação desta propriedade nos valores de atributo apresentados. Alguns exemplos onde a dependência funcional não irá existir: $A \rightarrow B$ ($A[a1]$); $B \rightarrow A$ ($B[b2]$); e $D \rightarrow C$ ($D[d3]$).

A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d2
a2	b2	c2	d3
a3	b3	c4	d3

Figura 8 – Relação R(A,B,C,D). Adaptado de Elmasri e Navathe (2011).

3.4.3 Normalização

Muitos dos projetos reais obtém projetos de bancos de dados em modelos legados ou de arquivos existentes. O processo de normalização tem como propósito apresentar um projeto de boa qualidade e que mantenha as características essenciais do projeto de base (ELMASRI; NAVATHE, 2011).

A normalização consiste em testes para verificar se um esquema de relações satisfaz certas restrições, ou formas normais. Caso não satisfaça a tais condições, é possível decompor a relação de modo que o esquema de relações se enquadre na forma normal proposta. Serão apresentadas neste trabalho três formas normais, nominadas primeira, segunda e terceira formas normais (ELMASRI; NAVATHE, 2011).

Primeira forma normal

Uma relação está na primeira forma normal (1FN) se todos os atributos da relação assumem valores atômicos. Ou seja, nenhum valor de atributo é divisível. Exemplos de atributos com valores não atômicos são os atributos compostos e os atributos multivalorado. Um atributo composto pode ser exemplificado por endereço, que possui cidade, bairro, logradouro e número da residência. Outro caso de atributo não atômico é um atributo que armazena uma lista de valores (SILBERSCHATZ; KORTH; SUDARSHAN, 2012).

Assuma a tabela PESSOA com adição da coluna certificação, representado pela nova relação PESSOA1, na Figura 9. Esta coluna representa os cursos ou seminários que a pessoa concluiu. Esta relação, no entanto, não está na primeira forma normal, pois o atributo certificação não é atômico, logo ele não é funcionalmente dependente da chave CPF (ELMASRI; NAVATHE, 2011).

PESSOA1

CPF	nome	idade	escolaridade	certificação
111.111.111-11	Maria	34	Mestrado	inglês, espanhol
222.222.222-22	Adriana	21	Ensino Fundamental	*null*
333.333.333-33	Marcos	45	Doutorado	inglês, francês, administração

Figura 9 – Relação PESSOA1.

Há três métodos de manipular a relação PESSOA1 de modo que ela se enquadre na primeira forma normal, segundo Elmasri e Navathe (2011):

1. Retirar o atributo certificação da relação PESSOA1 - criar uma nova relação, CERTIFICAÇÕES(CPF, certificação), para relacionar pessoa (através do atributo CPF) à uma certificação;

2. Replicar as tuplas da relação para cada valor de certificação - tomar {CPF, certificação} como chave e os demais valores (nome, idade, escolaridade) se repetem para o mesmo valor de CPF;

3. Criar uma coluna para cada certificação - caso o número máximo de valores seja conhecido, como "certificação1, certificação2, ..., certificaçãon". Entretanto, se a maioria das pessoas não têm a quantidade n de certificações, esta relação irá introduzir muitos valores *null* na base de dados.

A primeira abordagem é em geral a melhor escolha por não introduz redundância de informações como a segunda abordagem, nem gerar excesso de valores nulos como a terceira. Além disto, caso surgisse uma pessoa com uma nova certificação e a terceira abordagem estivesse sendo usada, seria necessário a adição de uma nova coluna, o que não ocorre ao se optar pela abordagem 1 (ELMASRI; NAVATHE, 2011).

Segunda forma normal

Uma relação está na segunda forma normal (2FN) se cada atributo que não faz parte de uma chave candidata, chamado de atributo não principal, for total e funcionalmente dependente da chave primária desta relação. Dependência funcional é total quando a dependência X → Y deixar de existir se algum atributo do conjunto X for retirado. Quando algum atributo de X pode ser removido sem comprometer a dependência X → Y, esta dependência é parcial (ELMASRI; NAVATHE, 2011).

Na relação EMPREGADO da Figura 7, onde a chave é {CPF, cargo}, todos os atributos apresentam dependência funcional parcial com a chave, visto que CPF → {nome, idade, escolaridade} e cargo → salario. Portanto esta relação está na primeira forma normal, mas não está na segunda (ELMASRI; NAVATHE, 2011).

O método para transformar a relação EMPREGADO para a segunda forma normal é particioná-la em relações menores, de forma que todos os atributos não principais das relações derivadas sejam totalmente dependentes funcionalmente da chave. As relações PESSOA e CARGO da Figura 2 mostram como ficaria a relação EMPREGADO após tal modificação (ELMASRI; NAVATHE, 2011).

Terceira forma normal

Uma relação está na terceira forma normal se ela está na segunda forma normal e não possui atributos não principais transitivamente dependentes da chave primária. Isto é, possuir uma dependência do tipo X → Y, onde X → Z e Z → Y quando o conjunto de atributos Z não é chave ou parte dela (ELMASRI; NAVATHE, 2011). De forma mais informal, quer dizer que os atributos não-chaves são independentes entre si, ou seja, nenhum desses atributos é funcionalmente dependente de qualquer combinação dos demais atributos não-chave (DATE, 2004).

Esta forma normal garante que a atualização de qualquer atributo não principal da relação será independente dos outros atributos desta relação (DATE, 2004). Esta forma normal também permite alcançar a diretriz 4 da Seção 3.4.1, já que uma operação de junção natural pelos atributos de Z resultariam em tuplas falsas (ELMASRI; NAVATHE, 2011).

A maneira de normalizar uma relação para a terceira forma normal é quebrá-la em relações menores. Esta separação deve ser feita para cada dependência transitiva encontrada, de modo que as relações resultantes tenham o conjunto Z, de atributos, como chave. Assim, a junção natural por Z irá recuperar a relação original sem gerar tuplas falsas (ELMASRI; NAVATHE, 2011).

3.5 Refatoração em banco de dados

Domingues (2014) propôs um processo de refatoração de banco de dados utilizando a BPMN - *Business Process Modeling Notation*. Em seu trabalho foram abordadas algumas críticas em relação ao processo de refatoração de banco de dados apresentado por Ambler e Sadalage (2006).

A BPMN é uma linguagem formal de representação que visa o mapeamento do processo. Um processo de negócio constitui um conjunto de atividades que atingem o objetivo do negócio ao serem realizadas. Estas atividades devem ter início e término bem definidos. Nesta notação é possível representar, além dos aspectos físicos do modelo do banco de dados, as interações do usuário, a documentação do processo e o retorno ao estado anterior à refatoração, se houverem erros ou resultados indesejáveis (DOMINGUES, 2014).

Ambler e Sadalage (2006) descrevem refatoração em banco de dados como uma forma de reestruturar o código em etapas. É importante ressaltar que este método mantém a semântica do sistema. Isto é, não há adição de novas funcionalidades ou dados novos durante a refatoração. As funcionalidades e os dados existentes tampouco são removidos. O resultado deste processo é o aperfeiçoamento da modelagem do código existente

(AMBLER; SADALAGE, 2006).

Refatorar um sistema é mais simples do que a refatoração do banco de dados, uma vez que o primeiro deve se preocupar apenas em manter seu comportamento semântico. Por outro lado, refatorar um banco de dados, além de ter que preservar o comportamento semântico, deve se levar em consideração também as aplicações que fazem uso desta base de dados (AMBLER; SADALAGE, 2006).

O cenário mais singelo é quando apenas uma aplicação utiliza esta base de dados. Neste caso, é possível refatorar o banco de dados e a aplicação simultaneamente. Esta alteração é discutida entre o administrador do banco de dados e o analista de sistemas para se certificarem de que a mudança é necessária. Então a refatoração é desenvolvida e testada no ambiente de desenvolvimento. Depois o código é integrado ao sistema e no caso de falhas nos testes, correções são feitas (AMBLER; SADALAGE, 2006).

Entretanto, quando há mais de uma aplicação utilizando a base de dados que passará pelo processo de refatoração, um período de transição se faz necessário. É um intervalo entre o banco de dados antes e depois da refatoração, onde a modelagem nova é colocada junto à antiga. Neste ponto as duas modelagens têm a capacidade armazenar os dados das aplicações. Durante este período de transição as aplicações devem se adaptar à nova modelagem do banco de dados (AMBLER; SADALAGE, 2006). A figura 10 ilustra este período.

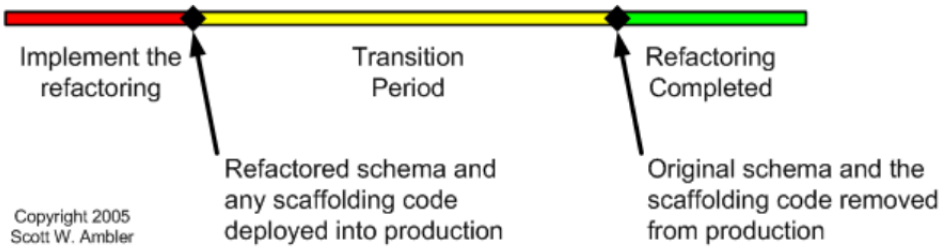

Figura 10 – Ciclo de vida de uma refatoração, segundo Ambler e Sadalage (2006). Retirado de Ambysoft (2017).

3.5.1 Manter a semântica

O significado e o comportamento dos esquemas de relações devem ser mantidos durante o processo de refatoração. Portanto, inclusão ou exclusão de dados não devem ser realizadas neste processo. Isto faz com que o usuário não seja afetado pelas mudanças feitas. Um exemplo é a refatoração que introduz um formato comum, como por exemplo refatorar os números de CPFs do formato “111.111.111-11” para “11111111111”. Apesar de ter sofrido modificação no banco de dados, o CPF ainda poderá ser acessado pelo usuário no formato antigo (AMBLER; SADALAGE, 2006).

A aplicação também deve se adaptar à refatoração do esquema de relação a fim de preservar a semântica. Por exemplo: uma refatoração introduz um método de cálculo e armazena este valor em uma coluna. Os métodos das aplicações que eram responsáveis por este cálculo agora devem se modificar para, ao invés de realizar o cálculo, consultar o valor na coluna apropriada do banco de dados (AMBLER; SADALAGE, 2006).

3.5.2 Categorias da refatoração de banco de dados

Ambler e Sadalage (2006) dividem a refatoração de banco de dados em seis categorias: estrutural, qualidade dos dados, integridade referencial, arquitetural, método e transformações não-refatoração. São apresentadas a seguir a descrição de cada uma delas e como elas se aplicam.

1. Estrutural: altera a definição de uma ou mais tabelas ou visões. Por exemplo, na transposição de uma coluna de uma tabela para outra; ou na fragmentação de uma coluna para várias.

2. Qualidade de dados: melhora a qualidade da informação contida no banco de dados. Esta categoria está presente na atribuição de restrição *not null* ou de formato válido em uma coluna.

3. Integridade referencial: mudança que garante que a coluna referenciada existe em outra tabela, ou que não haja referência para uma coluna sem necessidade. Exemplo desta categoria é adicionar uma *trigger* para deletar duas entidades em modo cascata.

4. Arquitetural: melhora o modo com o qual os programas externos interagem com o banco de dados. Esta categoria pode ser encontrada, por exemplo, na alteração de um método Java existente para uma biblioteca compartilhada com um resultado armazenado no banco de dados. Esta refatoração permite que aplicações que não são Java também tenham acesso à esta funcionalidade.

5. Método: alterações em procedimentos do banco de dados, como funções ou *triggers*, que agregam o fator de qualidade. Por exemplo, alterar o nome de um método a fim de facilitar seu entendimento.

6. Transformações não-refatoração: Mudanças na base de dados que alteram a semântica do esquema de relação. A criação de uma coluna em uma tabela do banco de dados se enquadra nesta categoria.

3.5.3 Indícios da necessidade de refatorar o banco de dados

Existem alguns indícios que indicam que o banco de dados necessita de um processo de refatoração. Ambler e Sadalage (2006) indicam os seguintes:

- Coluna com múltiplos propósitos: isto acontece quando uma coluna é usada para armazenar mais de um tipo de informação. Certamente haverá uma outra coluna, ou mais, nesta relação a fim de conduzir à interpretação correta destas informações.
- Tabela com múltiplos propósitos: similar ao item anterior, ocorre quando uma tabela representa mais de uma relação. Por exemplo, uma tabela VINCULO que armazena tanto as informações de funcionário, quanto as informações de estagiário. Haverá colunas para guardar dados como CURSO que será pertinente apenas para os estagiários. Desta forma, haverá colunas com valores nulos para alguns tipos de vínculos e para outros, não.
- Dados redundantes: a mesma informação armazenada em mais de um lugar dá abertura para ocorrência de inconsistência. Caso esta informação seja atualizada em apenas um desses campos, as aplicações que utilizam o outro campo terá a informação desatualizada.
- Tabelas com muitas colunas: este é mais um indício de que a tabela provavelmente representa mais de uma relação. Isto significa que não há coesão e que a tabela precisa passar por um processo de normalização a fim de separar corretamente as informações nela contidas.
- Tabelas com muitas linhas: este fator compromete a performance das consultas no banco de dados. Atribuir algumas colunas ou linhas para outra tabela pode minimizar este problema.
- Colunas “inteligentes”: a coluna “inteligente” é aquela que, de alguma forma, armazena mais de uma informação na mesma coluna. Por exemplo, no uso de máscara no campo MATRICULA para indicar a empresa daquela matrícula. A inicial da matrícula indica “U” ou “F” para identificar se a matrícula está associada à empresa “UFU” ou “FAEPU”, respectivamente.
- Medo de mudança: este fator está ligado ao medo, por exemplo, de que as aplicações que utilizam o banco de dados deixem de funcionar. Isto diz um pouco sobre como as informações das entidades estão acopladas. Este é um indício de que a mudança no esquema do banco de dados é necessária. Além disto, sugere que tal problema tende a piorar com o tempo.

3.5.4 Processo de refatoração

Algumas atividades do processo de refatoração de Ambler e Sadalage (2006), descrito na Figura 11, são desconsideradas no processo de refatoração de Domingues (2014). A justificativa para desconsiderá-las é listada na sequência.

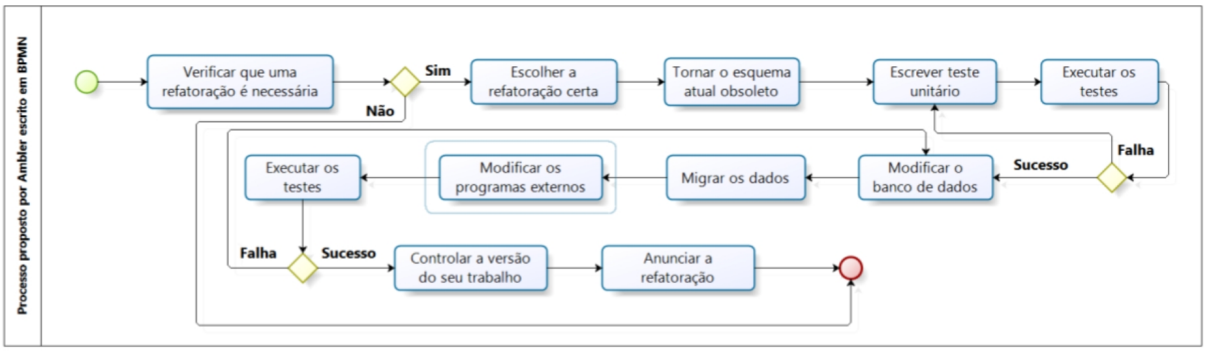

Figura 11 – Processo de refatoração proposto por Ambler e Sadalage (2006) escrito em BPMN. Retirado de Domingues (2014).

1. Verificar que uma refatoração é necessária: deveria fazer parte do contexto do processo e não do processo.

2. Escolher a refatoração certa: esta atividade pode limitar o processo, uma vez que pode ser necessário um conjunto de refatorações para conduzir o banco de dados a um estado correto.

3. Tornar o esquema obsoleto (opcional): tarefa considerada específica à empresa e como ela gerencia mudanças.

4. Escrever teste unitário e Executar os testes: não há ferramentas evoluídas para escrever testes unitários para banco de dados. Uma alternativa mais viável é executar um conjunto de *queries* e pontuar o teste com base no desempenho das consultas.

5. Modificar o banco de dados: o esquema original deveria ser guardado em todos os processos de refatoração, tanto no ambiente de produção quanto nos ambientes de desenvolvimento e de teste.

6. Migrar os dados: tarefa que não ocorre em todas as refatorações, por isso deveria ser uma parte da tarefa de executar refatoração.

7. Modificar os programas externos: esta é considerada uma atividade fora do escopo de um processo no banco de dados. Afinal, esta atividade sugere que o processo só

pode ser concluído caso a aplicação tenha se adaptado à refatoração do banco de dados.

8. Controlar a versão do seu trabalho: considera-se que o enfoque é um processo no banco de dados e que atualmente não há controle de versão. Por este motivo esta atividade é desconsiderada.

9. Anunciar a refatoração: esta tarefa também tem a ver com o contexto da refatoração, pois depende de como a organização lida com este processo. Além disto, só seria possível anunciar uma refatoração após o término do processo.

Em razão dos problemas levantados por Domingues (2014) no processo de Ambler e Sadalage (2006), fora criado um novo processo de refatoração de dados, ilustrado na Figura 12. Antes de iniciar o processo, é feita a coleta de requisitos dos usuários e dos fatores na modelagem que podem gerar problemas ou prejudicar o desempenho das consultas. Estes dados serão a entrada para o processo de refatoração (DOMINGUES, 2014).

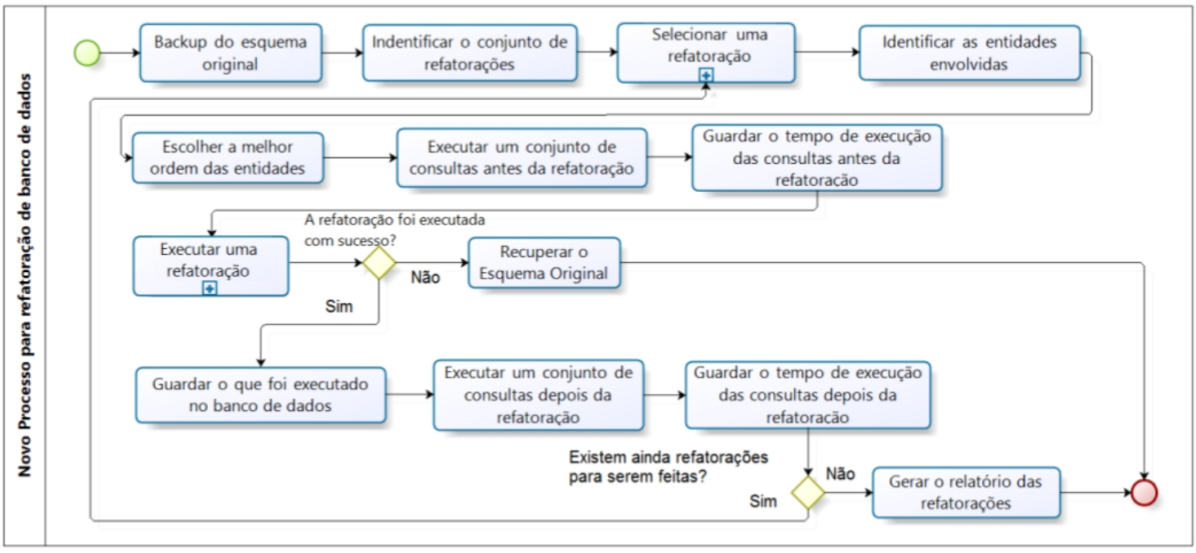

Figura 12 – Processo de refatoração proposto por Domingues (2014) escrito em BPMN. Retirado de Domingues (2014).

Este novo processo de refatoração de dados apresenta algumas vantagens em relação ao processo anterior (DOMINGUES, 2014):

- Considera a finalização do processo como a execução de um conjunto de refatorações;
- Cuida apenas das tarefas do administrador de banco de dados;
- Especifica o teste unitário como um conjunto de consultas;

- Coleta o tempo das consultas a fim de fazer análises sobre a melhora do desempenho após a refatoração;
- Encarrega a documentação da refatoração sobre a necessidade de migrar os dados, já que apenas uma minoria das refatorações possuem esta necessidade;
- A notação BPMN possibilita validação, avaliação, aperfeiçoamentos e automatização.
- Há a possibilidade de recuperar o esquema original em qualquer ambiente caso a refatoração não seja satisfatória.

Apesar de Domingues (2014) não considerar a atividade "Modificar os programas externos" como parte do processo de refatoração dos dados, considera-se que o processo de refatoração do código das aplicações pode ser realizado na sequência, se necessário. Assume-se que a forma adequada seria realizar a refatoração do código das aplicações após o processo de refatoração de dados no ambiente de desenvolvimento e antes deste mesmo processo no ambiente de produção (DOMINGUES, 2014).

4 Banco de Dados do SIAF

O Hospital de Clínicas Uberlândia (HCU) iniciou suas atividades em outubro de 1970 como unidade de ensino para parte do aprendizado aos cursos de Medicina da Escola, de Medicina e Cirurgia. Considerado o maior prestador de serviços pelo Sistema Único de Saúde (SUS) de Minas Gerais, é um importante centro de atendimento de urgências, emergências e de alta complexidade. Além disto, é o único hospital público da região com atendimento a todos os níveis de atenção à saúde disponível 24 horas por dia (UFU, 2017b).

Criado na década de 1980 no *mainframe* IBM 3090, o banco de dados do SIAF surgiu da necessidade de tornar públicas as escalas de trabalho dos profissionais do HCU. Inicialmente as mais de 200 variações de escalas de trabalho dos profissionais da saúde eram gerenciadas manualmente. Assim, o primeiro sistema desenvolvido para o hospital, do SIAF, foi o Escala.

O SIAF utilizava em sua primeira versão o banco de dados IBM DB2. Naquela época o *mainframe* processava todas as informações da universidade, tanto da área acadêmica quanto da área hospitalar. Com a evolução dos sistemas, o crescente nível de informatização e o progresso tecnológico, o banco de dados do SIAF passou a armazenar a folha de pagamento dos funcionários contratados pelo HCU e também a parte da gestão financeira geral do hospital com o *mainframe* IBM 9672.

Hoje, o banco de dados do SIAF é responsável por gerenciar as informações administrativas e financeiras dos contratados, vinculados à UFU, FAEPU ou a empresas terceirizadas, que prestam serviço ao hospital. Servidores UFU desempenham cargos como analista de sistema, biólogo, médico, entre outros. As pessoas contratadas por empresas terceirizadas podem assumir os cargos de educador físico, engenheiro e técnico em informática, por exemplo.

De acordo com o Estatuto da Fundação de Assistência, Estudo e Pesquisa de Uberlândia (FAEPU) (UFU, 2017a):

> Art. 4º A Fundação, sem finalidade lucrativa, tem por objetivos:[...] III - proporcionar à Universidade Federal de Uberlândia todo o apoio e os meios necessários à consecução dos seus objetivos, especialmente: [...] V - criar, instalar e manter ambulatórios e estabelecimentos hospitalares, para a prestação de serviços médicos, odontológicos e veterinários, remunerados ou gratuitos, atendendo à clientela própria ou de terceiros;

O SIAF é dividido em três grandes áreas: Materiais, Pessoas e Finanças, conforme Figura 13. A primeira área cuida dos materiais em consignação, artigos fixos (os patrimônios) e estoques. O setor de Finanças lida com a parte de contabilidade, orçamentos, dados bancários, centro de resultados, entre outros setores financeiros. O campo de Pessoas cuida de toda a parte de gestão de recursos humanos.

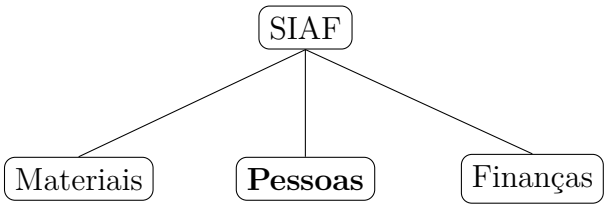

Figura 13 – Áreas que integram o sistema SIAF.

Dentro do segmento de Pessoas os principais sistemas são:

- Gestão RH - gerencia cadastros, admissões, demissões, transferências de funcionários e pagamento de plantões de pessoas vinculadas ao hospital;
- Avaliação de Desempenho FAEPU - detém informações sobre o desempenho no trabalho dos empregados da FAEPU;
- Escala - coordena a escala de trabalho das pessoas que trabalham para o hospital;
- Meu Ponto - registra ocorrências de batida de ponto dos empregados;
- Apuração de ponto - controla informações cruzadas do sistema Escala com a batida de ponto do sistema Meu Ponto.

Com quase quatro décadas ativo, o banco de dados do SIAF possui atualmente mais de 500 tabelas para as áreas de Materiais, Pessoas e Finanças. Por este motivo, optou-se por restringir o escopo utilizado no desenvolvimento deste trabalho ao subsistema do sistema SIAF, da áreaa de Pessoas, Gestão RH e às tabelas utilizadas por ele.

4.1 Sistema Gestão RH

O módulo Gestão RH gerencia os dados dos recursos humanos vinculados ao HCU, bem como as informações que especificam o tipo de vínculo com o hospital. Atua sobre a arquitetura Intel e Linux, mantém o banco IBM DB2 na versão 8 com previsão de atualização para a versão 11 até o final deste ano.

Na década de 90, por questão de custos de sistemas, iniciou-se a fase de *downsizing* das plataformas da universidade e passou-se a utilizar o sistema AIX (Unix) na plataforma

RISC. O último módulo vinculado ao gerenciamento de recursos humanos a ser migrado para a plataforma RISC no processo de *downsizing* foi o Folha de Pagamento em virtude de sua criticidade.

Devido ao curto prazo no cronograma, este módulo fora migrado sem remodelagem do banco ou revisão prévia dos processos. Assim, o Gestão RH foi desenvolvido sem integração com o sistema Folha de Pagamentos, ou com sistemas terceirizados (Sankhya, DMP Access, Thales Group).

O fator mais impactante, porém, foi a falta de uma base de dados única para o sistema de recursos humanos. Em função disto, surgiram numerosos sistemas de pequeno porte que continham cadastro de empregados e acadêmicos em bancos paralelos (a maioria *Firebird*).

Entretanto, quando o módulo Gestão RH fora implantado, os demais sistemas com gerenciamento de empregados começaram a ser incorporados a ele. Assim, o módulo Gestão RH consolidou-se como o sistema responsável pela gestão de pessoas e as bases de dados paralelas foram unificadas ao banco de dados do SIAF. Logo as bases de dados paralelas foram desativadas e o base de dados do SIAF firmou-se como a base de dados única para armazenamento de informações de recursos humanos do hospital.

4.1.1 Tabelas

- PESSOARH - armazena as informações pessoais dos empregados vinculados ao HCU, como dados do documento de identidade e informações de contato. A chave primária da relação é PESCPF (CPF da pessoa), que também é chave estrangeira na relação PESSOATIPORH.
- PESSOATIPORH - guarda os dados relativos ao tipo de pessoa vinculada ao hospital e as informações específicas deste tipo. Pode conter os tipos funcionário, estagiário, prestador de serviço, didático, representante, requisitado e aprendiz.
- CENTROCUSTO - define as unidades do HCU responsáveis por manejar seus recursos financeiros de forma autônoma.
- FUNCENTROCUSTO - registra a ligação entre cada registro de PESSOATIPORH e o CENTROCUSTO responsável por gerenciá-lo.
- FUNCAORH - armazena dados das pessoas que possuem, além do vínculo empregatício registrado na tabala FUNCENTROCUSTO (com centro de custo responsável e cargo), alguma função a além, como gerente ou diretor.

5 Desenvolvimento

No desenvolvimento deste projeto foi realizado um dos processos da reengenharia de *software* (Seção 3.1.2), a reengenharia de dados. Este processo se deu através da aplicação das regras de normalização apresentadas na Seção 3.4.3, seguindo também as diretrizes de projeto de banco de dados vistas na Seção 3.4.1. A remodelagem tem como propósito a reestruturação dos dados, ou seja, a revisão e análise do agrupamento dos atributos em função da relação a qual eles pertencem.

Devido ao fato de que o processo real de reengenharia de *software* e seus subprocessos demandarem grande investimento financeiro, tempo e treinamento de muitas pessoas (SILBERSCHATZ; KORTH; SUDARSHAN, 2012), a remodelagem deste trabalho foi feita apenas para o tipo de vínculo "funcionário", da tabela PESSOATIPORH. A Figura 14 ilustra o escopo que será utilizado no desenvolvimento.

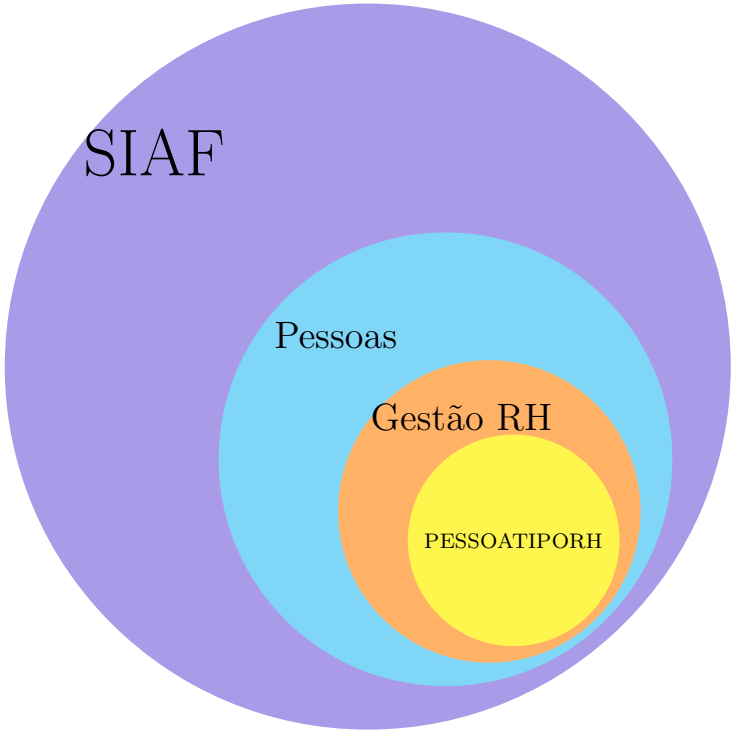

Figura 14 – Escopo do desenvolvimento deste trabalho.

5.1 Relação PESSOATIPORH

A diversidade de tipos de vínculo que um colaborador contratado pelo hospital pode pode assumir, registrado na tabela PESSOATIPORH, é responsável pela maior parte dos valores nulos encontrados nesta tabela. Isto acontece porque cada um dos tipos

demanda um conjunto particular de atributos, enquanto a tabela contempla o conjunto de cada um deles. De forma detalhada, tem-se as seguintes categorias de tipos:

1. Funcionário
2. Estagiário
3. Voluntário
4. Prestador de Serviços
5. Didático
   - 5.1. Interno
   - 5.2. Acadêmico
   - 5.3. Residente
   - 5.4. Professor de 3º Grau
   - 5.5. Pós Graduação
   - 5.6. Educação Médica Continuada
   - 5.7. Escola de Governo
6. Representante
7. Requisitado
8. Aprendiz

Atualmente o banco de dados do SIAF armazena 16622 tuplas da relação PESSOATIPORH. A distribuição dos tipos de vínculo que ela possui são indicados no gráfico de pizza da Figura 15.

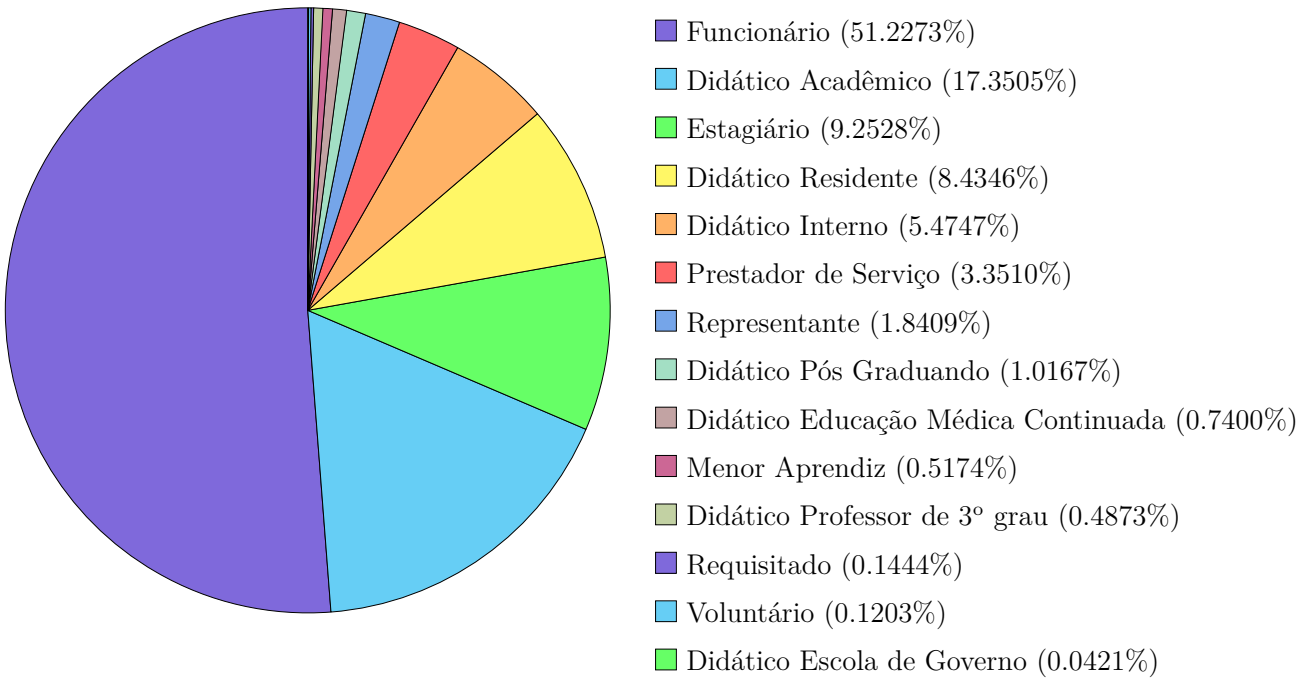

Figura 15 – Distribuição dos tipos de vínculos da tabela PESSOATIPORH.

Para representar todos estes tipos de pessoas e suas variantes, a tabela PESSOATIPORH possui 51 colunas. Todos estes atributos são listados a seguir.

1. PESSOATIPOID
2. PESCPF
3. EMPRESAID
4. PTCODIGO
5. MATRICULA
6. DIGITOV
7. PTDATAINICIO
8. PTDATATERMINO
9. PTREGCONSELHO
10. PTAGENCIA
11. PTBANCO
12. PTCONTACORRENTE
13. PTGRADUACAO
14. PTCOORDENADOR
15. PTDIRETORCURSO
16. PTSUPERVISOR
17. PTCHEFIA
18. PTESTAGIO
19. REGIMETRABALHOID
20. PTBOLSISTA
21. PTFUNCAO
22. PTOBRIGATORIEDADE
23. PTCURSO
24. PTVALORBOLSA
25. PTPERIODO
26. PTPOSGRADUACAO
27. PTPISPASEP
28. PTCENTROCUSTOID
29. PTCARGAHORARIA
30. PTCONTRIBUINTE
31. PTESPECIALIDADE
32. PTEMPREPRESENTANTE
33. PTESPEC
34. PTSTATUS
35. PTPLANTONISTA
36. PTNOMETERCEIRA
37. PESIMPORTACAO
38. USUARIOCAD
39. DATAHORACAD
40. USUARIOALTER
41. DATAHORAALTER
42. PTORIGEM
43. PTTIPOESTAGIO
44. PTSUPERVISORAREA
45. PTDATARESCISAO
46. PTCODDESLIGAMENTO
47. PTPERIODOSTR
48. PTPESSOATIPOREF
49. PTRAIOX
50. PTAVALIADOR
51. CONSELHOID

A chave para esta relação é {PESSOATIPOID, PESCPF, EMPRESAID, PTCODIGO}. O atributo PESSOATIPOID designa um dos 14 tipos de vínculo já mencionados. A lista dos tipos é encontrada na tabela CODIFICAÇAO. PESCPF identifica a tupla em PESSOARH a qual pessoa este registro pertence. EMPRESAID indica a empresa, listado na relação EMPRESAFLH.

O atributo PTCODIGO é um valor incremental utilizado para caracterizar uni-

camente um registro. Caso uma pessoa algum dia já tenha sido registrada nesta tabela com os mesmos valores dos demais atributos da chave, este valor é usado. Isto acontece quando um empregado é desligado e volta a ser contratado pelo hospital.

5.2 Identificação de valores nulos na tabela PESSOATIPORH

Dos 51 atributos da tabela PESSOATIPORH, há aqueles que são comuns a todos os tipos de vínculos como os atributos de controle USUARIOCAD, DATAHORACAD, USUARIOALTER e DATAHORAALTER. Entretanto, a maioria das colunas é composta por atributos específicos a apenas um tipo, como DIGITOV para servidores UFU (funcionário contratado pela UFU) e PTCOORDENADOR para estagiário.

É possível notar, então, que determinados atributos da relação PESSOATIPORH, quando analisados em relação a um determinado tipo de vínculo, não agregam nenhum valor semântico a este vínculo. Este fator viola a primeira diretriz do projeto de banco de dados, que diz respeito à clareza da semântica dos atributos na relação.

É fácil perceber, por exemplo, que os atributos PTDIRETORCURSO e PTVALORBOLSA são exclusivos do vínculo estagiário, já que são informações pertinentes ao contrato de estágio. Portanto, estes atributos não contribuem em nada ao significado dos demais vínculos da relação PESSOATIPORH.

Desta forma, estes atributos, quando registrados para algum tipo de vínculo no qual não agregarão nenhum valor ao registro, sempre receberão o valor *null* nas colunas correspondentes a estes atributos. Em virtude deste fato, a terceira diretriz do projeto de banco de dados que trata de redução de valores nulos também é violada.

Na sequência é apresentada, novamente, a lista de todas as colunas da relação PESSOATIPORH, mas deste vez considerando o vínculo do tipo “funcionário”. Os atributos destacados em vermelho representam as colunas que sempre receberão valor nulo por não fazerem sentido algum para o vínculo do tipo “funcionário”.

1. PESSOATIPOID
2. PESCPF
3. EMPRESAID
4. PTCODIGO
5. MATRICULA
6. DIGITOV
7. PTDATAINICIO
8. PTDATATERMINO
9. PTREGCONSELHO
10. PTAGENCIA
11. PTBANCO
12. PTCONTACORRENTE
13. PTGRADUACAO
14. PTCOORDENADOR
15. PTDIRETORCURSO
16. PTSUPERVISOR
17. PTCHEFIA
18. PTESTAGIO
19. REGIMETRABALHOID
20. PTBOLSISTA
21. PTFUNCAO
22. PTOBRIGATORIEDADE
23. PTCURSO
24. PTVALORBOLSA
25. PTPERIODO
26. PTPOSGRADUACAO
27. PTPISPASEP
28. PTCENTROCUSTOID
29. PTCARGAHORARIA
30. PTCONTRIBUINTE
31. PTESPECIALIDADE
32. PTEMPREPRESENTANTE
33. PTESPEC
34. PTSTATUS
35. PTPLANTONISTA
36. PTNOMETERCEIRA
37. PESIMPORTACAO
38. USUARIOCAD
39. DATAHORACAD
40. USUARIOALTER
41. DATAHORAALTER
42. PTORIGEM
43. PTTIPOESTAGIO
44. PTSUPERVISORAREA
45. PTDATARESCISAO
46. PTCODDESLIGAMENTO
47. PTPERIODOSTR
48. PTPESSOATIPOREF
49. PTRAIOX
50. PTAVALIADOR
51. CONSELHOID

Como apresentado na Seção 3.3, estes nulos que ocorrem em uma coluna inteira da tabela PESSOATIPORH em todos os registros de um determinado tipo de vínculo não acontecem por se tratarem de uma informação desconhecida, ou propositalmente não informada (indisponível). Estes nulos, no entanto, são claramente da categoria de nulos de valores não aplicáveis, uma vez que estes atributos não se aplicam a alguns tipos de vínculo.

5.3 Tratamentos dos valores nulos identificados

Os atributos da tabela PESSOATIPORH serão reagrupados para o tipo funcionário. Este conjunto de atributos irá compor uma nova tabela da PESSOATIPORH, que será chamada a partir de agora neste trabalho como tabela FUNCIONARIO. O objetivo esperado é que este reagrupamento dos atributos na tabela FUNCIONARIO tenham uma semântica clara e que sejam reduzidos os valores nulos.

1. Lista de atributos da tabela FUNCIONARIO

    1.1. PESCPF
    1.2. EMPRESAID
    1.3. PTCODIGO
    1.4. MATRICULA
    1.5. DIGITOV
    1.6. PTDATAINICIO
    1.7. PTDATATERMINO
    1.8. PTREGCONSELHO
    1.9. REGIMETRABALHOID
    1.10. PTCARGAHORARIA
    1.11. PTPISPASEP
    1.12. PTSTATUS
    1.13. USUARIOCAD
    1.14. DATAHORACAD
    1.15. USUARIOALTER
    1.16. DATAHORAALTER
    1.17. PTDATARESCISAO
    1.18. PTCODDESLIGAMENTO
    1.19. CONSELHOID

Como mostrado no gráfico da Figura 15, o tipo funcionário ocupa 51.23% dos registros da tabela PESSOATIPORH. Após reagrupar apenas os atributos pertinentes à relação FUNCIONÁRIO, houve a redução de 51 para 18 atributos. Ou seja, a redução de mais de 60% dos atributos de PESSOATIPORH para as tuplas do tipo funcionário.

Sabe-se que as colunas desconsideradas na lista de atributos da tabela FUNCIONARIO possuem somente nulos do tipo não aplicáveis. Desta forma, utilizando apenas a análise semântica dos atributos, aplicando a primeira diretriz do projeto de banco de dados, a relação FUNCIONARIO já reduz 32% dos valores nulos da tabela PESSOATIPORH.

Observe que um dos atributos da chave também foi retirado da relação FUNCIONARIO. O atributo removido, PESSOATIPOID, identifica o tipo de vínculo que a relação PESSOATIPORH possui. Entretanto, agora é possível referenciar as instâncias do tipo de vínculo "funcionário" através da própria tabela FUNCIONARIO. Por isso, o atributo PESSOATIPOID não é necessário nesta nova relação.

Normalização

Serão aplicadas na relação FUNCIONARIO as regras da primeira, segunda e terceira formas normais no processo de normalização de dados. Para isto, cada atributo

deve ser analisado individualmente e em relação à chave {PESCPF, EMPRESAID, PTCODIGO}.

- MATRICULA - valor único para cada FUNCIONARIO na empresa;
- DIGITOV - dígito verificador do campo MATRICULA das tuplas da tabela FUNCIONARIO quando a empresa é UFU;
- PTDATAINICIO - data de início do contrato de trabalho do FUNCIONARIO;
- PTDATATERMINO - data de fim do contrato de trabalho do FUNCIONARIO;
- PTREGCONSELHO - número do registro de (algum) conselho;
- REGIMETRABALHOID - chave estrangeira para tabela REGIMETRABALHO, que armazena todos os regimes de trabalhos possíveis;
- PTPISPASEP - número do PIS ou do PASEP do FUNCIONARIO;
- PTCARGAHORARIA - guarda a carga horária semanal de trabalho do FUNCIONARIO;
- PTSTATUS - mostra se o FUNCIONARIO exerce alguma atividade no HCU no momento. Armazena o código dos valores ATIVO ou INATIVO, obtidos na tabela CODIFICACAO;
- USUARIOCAD - usuário responsável pelo cadastro deste registro;
- DATAHORACAD - data do cadastro deste registro;
- USUARIOALTER - usuário responsável pela alteração deste registro;
- DATAHORAALTER - data de alteração deste registro;
- PTDATARESCISAO - data de encerramento do contrato de vínculo empregatício do FUNCIONARIO;
- PTCODDESLIGAMENTO - chave estrangeira para a tabela CODIFICACAO, que possui os motivos possíveis para o desligamento do FUNCIONARIO;
- CONSELHOID - chave estrangeira para a tabela CONSELHOPROFISSIONAL, que guarda informações dos conselhos registrados.

1FN

Os atributos da relação FUNCIONARIO, bem como da tabela PESSOATIPORH, já estão na primeira forma normal. Todos os atributos são atômicos (simples e indivisíveis). Portanto, a relação não é modificada.

2FN

Na segunda forma normal, todos os atributos não principais devem ter dependência total com a chave. Neste caso, a dependência é parcial se algum atributo da chave {PESCPF, EMPRESAID, PTCODIGO} puder ser retirado sem causar prejuízo à dependência dos atributos.

Os atributos PTREGCONSELHO, PTPISPASEP e CONSELHOID dependem apenas de um dos atributos da chave, o atributo PESCPF. Estes atributos estão relacionados, respectivamente, ao número de registro do conselho profissional, ao número do PIS ou do PASEP e ao identificador do conselho. A Figura 16 mostra estas dependências no esquema de relação parcial de FUNCIONARIO.

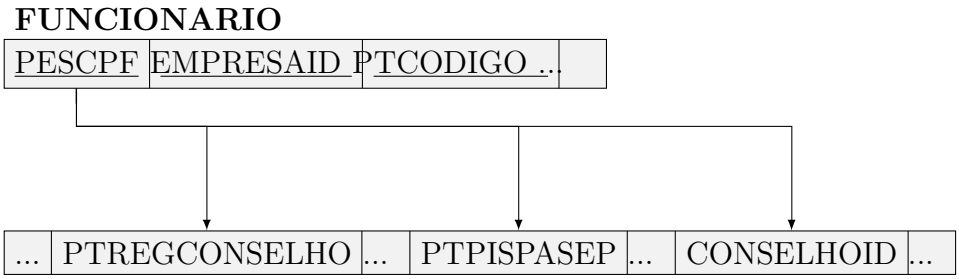

Figura 16 – Dependência parcial com a chave.

Os atributos EMPRESAID e PTCODIGO também fazem parte da chave de FUNCIONARIO. Entretanto, os atributos PTREGCONSELHO, PTPISPASEP e CONSELHOID independem da empresa na qual o FUNCIONARIO presta serviço, ou do número do contrato empregatício. Por este motivo estes atributos devem ser removidos desta relação.

Como cada pessoa pode ter apenas um número de PIS/PASEP, ele pode ser agrupado à relação PESSOARH existente, que reúne os dados pessoais, como documentos de identificação e informações de contato. A Figura 17 ilustra a adição deste atributo à tabela PESSOARH.

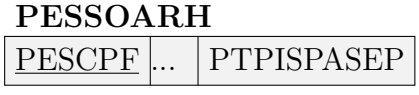

Figura 17 – Relação PESSOARH.

Os atributos excedentes PTREGCONSELHO e CONSELHOID carecem de uma nova relação, chamada, por exemplo, PESSOACONSELHO. A tabela PESSOACONSELHO, como ilustrada na Figura 18, irá conter as chaves estrangeiras PESCPF e CONSELHOID como chave. PESCPF identifica a pessoa que possui PTREGCONSELHO (o número de registro do conselho), e CONSELHOID irá identificar a qual conselho profissional este registro se refere.

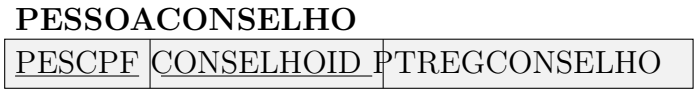

Figura 18 – Relação PESSOACONSELHO.

2. Lista de atributos da tabela FUNCIONARIO na 2FN

2.1. PESCPF
2.2. EMPRESAID
2.3. PTCODIGO
2.4. MATRICULA
2.5. DIGITOV (UFU)
2.6. PTDATAINICIO
2.7. PTDATATERMINO
2.8. REGIMETRABALHOID
2.9. PTCARGAHORARIA
2.10. PTSTATUS
2.11. USUARIOCAD
2.12. DATAHORACAD
2.13. USUARIOALTER
2.14. DATAHORAALTER
2.15. PTDATARESCISAO
2.16. PTCODDESLIGAMENTO

3FN

Para se enquadrar na terceira forma normal, a relação não pode ter dependência transitiva dos atributos não principais com a chave primária. Na tabela FUNCIONARIO, no entanto, pode ser observada a dependência transitiva entre o atributo DIGITOV e a chave através do campo MATRICULA, como mostra a Figura 19.

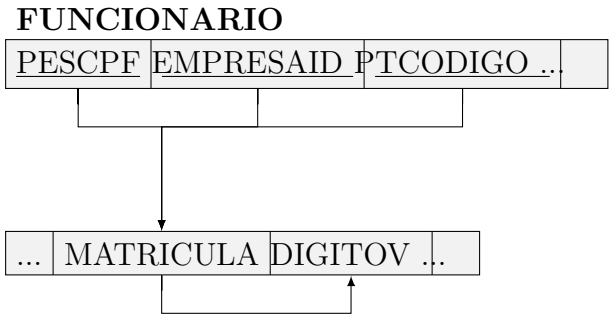

Figura 19 – Dependência transitiva na relação FUNCIONARIO.

A forma de resolver esta dependência transitiva é criar uma nova relação, MATRICULAUFU, Figura 20. Esta tabela irá armazenar apenas matrículas e dígitos verificadores de funcionários da UFU, pois é quando o atributo DIGITOV se aplica. Esta relação terá o atributo MATRICULA como chave, uma vez que a matrícula é única para cada funcionário na empresa.

Pode-se notar ainda a dependência transitiva do atributo REGIMETRABALHOID para PTCARGAHORARIA e do atributo PTDATARESCISAO para PTSTATUS. O primeiro corresponde ao campo REGHORASSEMANA da tabela REGIMETRABALHO. E o atributo STATUS é um atributo derivado de PTDATARESCISAO.

MATRICULAUFU

MATRICULA	DIGITOV

Figura 20 – Relação MATRICULAUFU.

Estas dependências transitivas, no entanto, são resolvidas de forma diferente da dependência encontrada no atributo DIGITOV. Isto porque a informação do atributo PTCARGAHORARIA pode ser consultada na tabela REGIMETRABALHO através do campo REGIMETRABALHOID que já está na relação FUNCIONARIO e STATUS pode ser calculado de acordo com a data do campo PTDATARESCISAO e a data corrente.

Desta forma, estes campos não devem ser armazenados no banco, mas sim consultados ou calculados quando necessário (SILBERSCHATZ; KORTH; SUDARSHAN, 2012). Assim, a diretriz 2 do projeto de bancos de dados pôde ser alcançada, uma vez que houve redução de informações redundantes nas tuplas da relação FUNCIONARIO.

3. Lista de atributos da tabela FUNCIONARIO na 3FN

3.1. PESCPF
3.2. EMPRESAID
3.3. PTCODIGO
3.4. MATRICULA
3.5. PTDATAINICIO
3.6. PTDATATERMINO
3.7. REGIMETRABALHOID
3.8. USUARIOCAD
3.9. DATAHORACAD
3.10. USUARIOALTER
3.11. DATAHORAALTER
3.12. PTDATARESCISAO
3.13. PTCODDESLIGAMENTO

5.4 Análise do desenvolvimento

Nas Figuras 21 e 22, são apresentados os diagramas de entidade-relacionamento das relações PESSOATIPORH e FUNCIONARIO, respectivamente. As tabelas que possuem ligação com estas relações também foram representadas com o conjunto reduzido dos atributos. Isto porque o intuito dessas figuras é enfatizar as diferenças entre as tabelas utilizadas no desenvolvimento deste trabalho. A notação utilizada no diagrama é a Notação *Crow's Foot*. O significado dos terminais das linhas pode ser encontrado no canto inferior direito da Figura 21, diagrama entidade-relacionamento da relação PESSOATIPORH.

É possível observar que o diagrama de PESSOATIPORH (Figura 21) possui seis tabelas, enquanto o diagrama de FUNCIONARIO (Figura 22) possui uma relação a mais. Também é possível notar que o terminal da tabela REGIMETRABALHO está representado de forma diferente nos diagramas de PESSOATIPORH e FUNCIONARIO.

A terminação da linha da tabela REGIMETRABALHO do diagrama da Figura 21 indica "zero ou um", mesmo que este atributo faça parte do conjunto semântico do vínculo funcionário. Entretanto, esta condição não pode ser colocada na relação PESSOATIPORH pois ela é uma tabela genérica, com o propósito de representar vários tipos de vínculos e nem todos possuem REGIMETRABALHO. Diferentemente da relação PESSOATIPORH, a relação FUNCIONARIO está livre para impor esta condição, bem como todas as outras pertinente à entidade Funcionário.

Suponha o seguinte cenário: uma pessoa irá se cadastrar no GDHS para iniciar o vínculo empregatício no HCU e possui dois registros de conselhos, o Conselho Federal de Enfermagem (COFEN) e o Conselho Federal de Administração (CFA). Considerando a modelagem atual (Figura 21), o contratado deve escolher apenas um de seus registros para armazenar no banco de dados. Porém, a modelagem proposta (Figura 22) dá suporte para a inclusão de todos os registros de conselho que a pessoa possui.

Portanto, apesar da remodelagem da relação PESSOATIPORH resultar em mais tabelas, este fator não indica uma modelagem de dados ruim. Pelo contrário; o aumento na quantidade de relações permite que cada relação exerça uma responsabilidade específica e torne o esquema de relações muito mais coeso.

5.5 Generalização do desenvolvimento

O método utilizado no desenvolvimento deste capítulo, que resultou na relação FUNCIONARIO a partir da relação PESSOATIPORH deve ser aplicado também aos demais tipos de vínculos. Assim, a relação PESSOATIPORH irá originar várias relações individuais, cada uma contendo o conjunto de atributos que lhe será semanticamente pertinente.

É claro que é necessário fazer uma análise mais detalhada de cada tipo de vínculo e da semântica que a relação irá desempenhar em todo o esquema de relações. Entretanto, não faz parte do escopo deste trabalho entrar em detalhes de cada um dos tipos de vínculos de PESSOATIPORH. Por isso, é apresentado na Figura 23 como ficaria, de maneira geral, a disposição dessas relações no diagrama entidade-relação após a remodelagem dos tipos de vínculo de PESSOATIPORH.

Os seis tipos que ocupam menos de 1% das tuplas da tabela PESSOATIPORH foram desconsiderados no diagrama da Figura 23. Juntos eles representam cerca de 2% dos registros da relação PESSOATIPORH e a probabilidade de terem entrado em desuso no sistema é grande. Mas, como o intuito do diagrama das relações é dar uma ideia de como ficariam as relações após a remodelagem da tabela PESSOATIPORH, nada impede que uma relação não representada no diagrama exista no esquema de relações real.

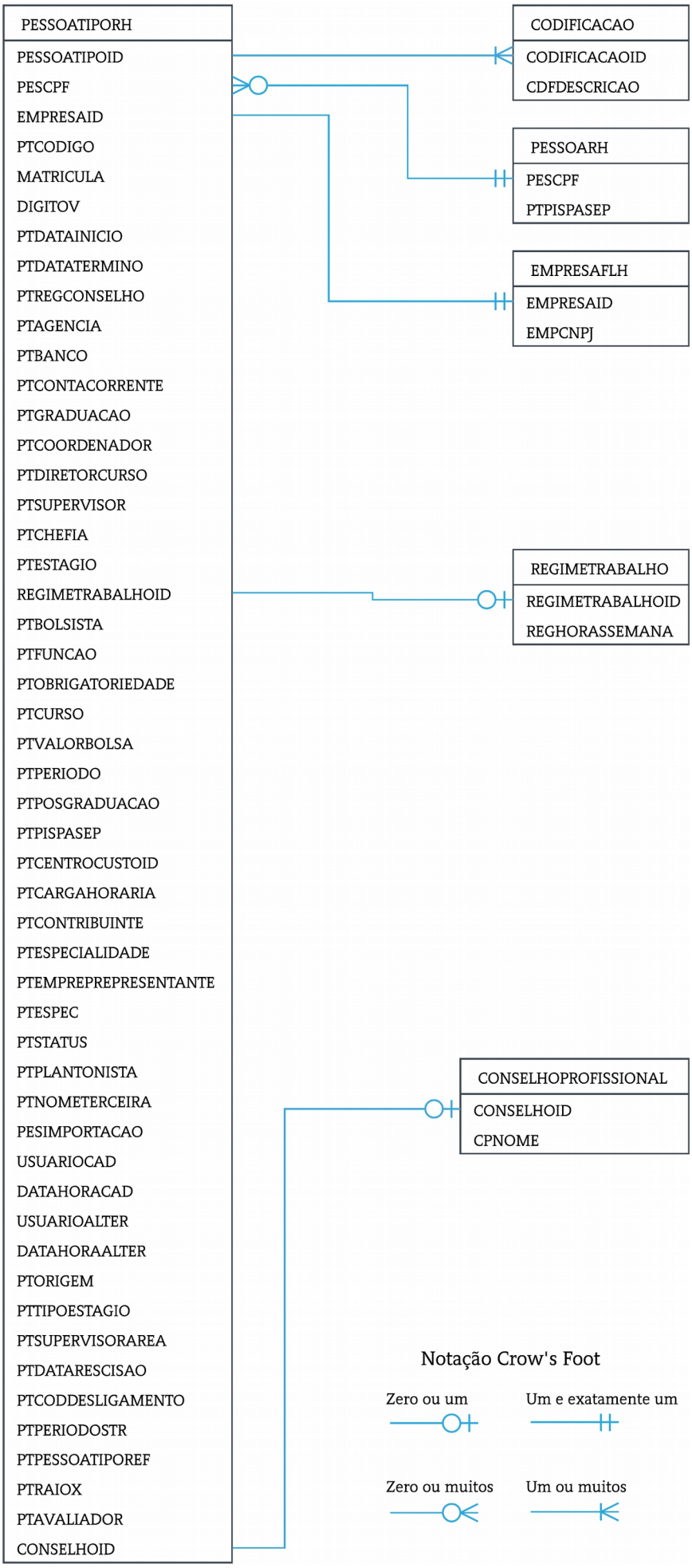

Figura 21 – Diagrama entidade-relacionamento PESSOATIPORH.

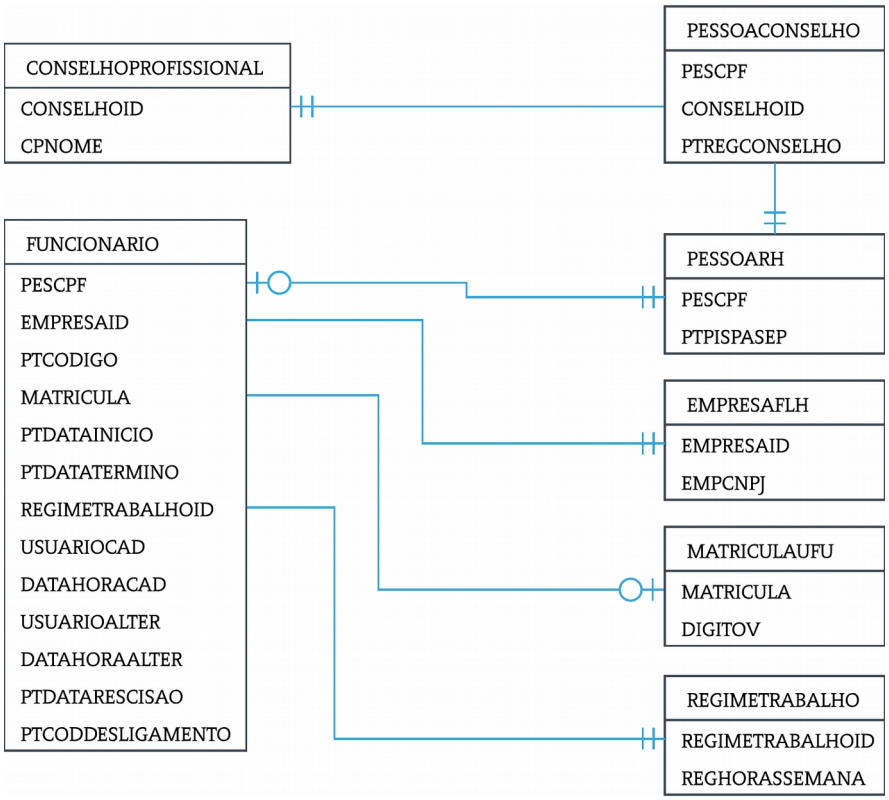

Figura 22 – Diagrama entidade-relacionamento FUNCIONARIO.

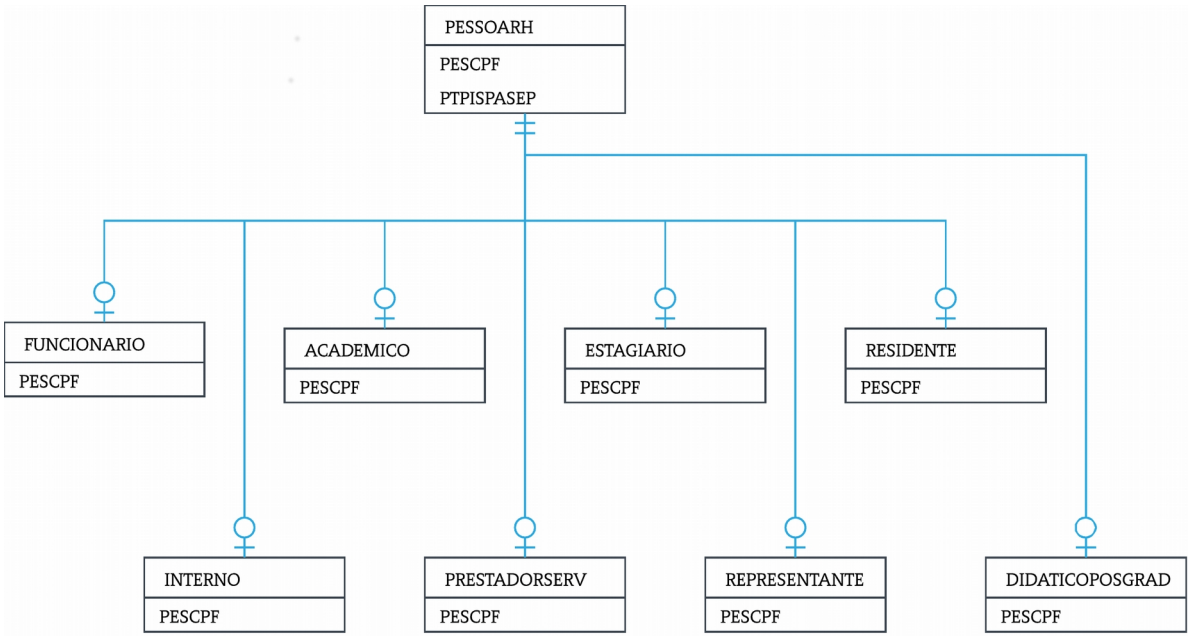

Figura 23 – Diagrama entidade-relacionamento das relações.

6 Conclusão e Trabalhos Futuros

6.1 Conclusão

O desenvolvimento do presente trabalho aplicou as regras de normalização da primeira à terceira forma normal na relação PESSOATIPORH a partir da perspectiva semântica do tipo de vínculo "funcionário". O tipo de vínculo escolhido para a técnica de refatoração estrutural foi suficiente para demonstrar a aplicação de cada regra de normalização, bem como das diretrizes de projeto de banco de dados.

Acredita-se fortemente que as situações encontradas nesta entidade estão presentes também nas demais entidades do banco de dados do SIAF. Em virtude disto, as conclusões obtidas na refatoração estrutural da tabela PESSOATIPORH serão generalizadas para as outras tabelas do banco de dados do SIAF.

O método utilizado na refatoração estrutural começa extraindo a relação FUNCIONARIO dos tipos de vínculos da tabela PESSOATIPORH. Ao reagrupar apenas os atributos pertinentes ao tipo FUNCIONARIO, houve a redução de mais de 60% dos atributos da relação original PESSOATIPORH. Este passo não promoveu apenas a **redução dos valores nulos**, mas também a **semântica clara no esquema**.

Além de também contribuir para a **semântica clara no esquema**, a mudança do atributo PTPISPASEP para a entidade PESSOARH, na segunda forma normal, também colaborou para a **redução de informações redundantes**. Isto porque cada pessoa pode assumir mais de um tipo de vínculo na empresa e há apenas um valor PISPASEP por pessoa.

Outra vantagem originada da 2FN foi a **possibilidade de representar certas informações** ao criar a tabela PESSOACONSELHO, onde podem ser atribuídos mais de um registro de conselho de cada pessoa. Sem contar com a **redução de valores nulos** nos casos em que a pessoa não possui nenhum registro de conselho.

A terceira forma normal também promoveu a **semântica clara no esquema** e a **redução dos valores nulos** no banco de dados do SIAF quando criou a nova tabela MATRICULAUFU. Desta forma, os valores dos dígitos verificadores (DIGITOV) das matrículas dos servidores UFU serão armazenados nesta tabela do banco de dados e não haverá este campo com valor *null* para as tuplas onde este valor não se aplica (funcionário da FAEPU e de outras instituições).

O SIAF é um sistema legado de quase cinco décadas e tem armazenado em sua base de dados informações primordiais à gerência do HCU ao longo desses anos. É importante

considerar que o processo de refatoração no banco de dados do SIAF implicaria também na refatoração do próprio sistema. Desta forma, refatorar a base de dados do SIAF requer a mobilização das equipes da organização: analistas de Banco de Dados e de Sistemas.

6.2 Trabalhos Futuros

O desenvolvimento apresentado no Capítulo 5 deste trabalho é apenas uma das atividades do processo completo de refatoração de banco de dados, especificamente, uma refatoração estrutural de dados. Desta forma, deve-se realizar as demais tarefas do processo de refatoração de banco de dados, bem como as refatorações das demais categorias (Seção 3.5.2), se necessário. Tanto a proposta tradicional do processo de refatoração de Ambler e Sadalage (2006) quanto a proposta de um novo processo de Domingues (2014) descritas na Seção 3.5 podem ser seguidas para efetivar a refatoração do banco de dados do SIAF.

Referências

AMBLER, S. W.; SADALAGE, P. J. *Refactoring Databases: Evolutionary Database Design*. New York: Addison-Wesley Professional, 2006. Citado 10 vezes nas páginas 6, 10, 16, 36, 37, 38, 39, 40, 41 e 60.

AMBYSOFT. *The Process of Database Refactoring: Strategies for Improvingo Database Quality*. 2017. Disponível em: <http://www.agiledata.org/essays/databaseRefactoring.html>. Acesso em: 04 dez. 2017. Citado 2 vezes nas páginas 6 e 37.

AQUAFOLD. *Aqua Data Studio*. 2017. Disponível em: <http://www.aquafold.com/aquadatastudio>. Acesso em: 5 dez. 2017. Citado na página 15.

DATE, C. J. *Introdução a sistemas de bancos de dados*. 8. ed. Rio de Janeiro: Elsevier, 2004. Citado 10 vezes nas páginas 6, 15, 16, 22, 24, 26, 27, 28, 29 e 36.

DOMINGUES, M. B. P. *Um novo processo para refatoração de bancos de dados*. Tese (Doutorado) — Escola Politécnica, Universidade de São Paulo, São Paulo, 2014. doi:<10.11606/T.3.2014.tde-29122014-165740>. Acesso em: 2 dez. 2017. Citado 7 vezes nas páginas 6, 16, 36, 40, 41, 42 e 60.

ELMASRI, R.; NAVATHE, S. B. *Sistemas de banco de dados*. 6. ed. São Paulo: Addison-Wesley, 2011. Citado 15 vezes nas páginas 6, 10, 11, 15, 16, 26, 27, 28, 30, 31, 32, 33, 34, 35 e 36.

LIENTZ, B. P.; SWANSON, E. B. *Software Maintanence Management*. Reading (Massachusetts): Addison-Wesley, 1980. Citado na página 19.

OSBORNE, W. M.; CHIKOFSKY, E. J. *Fitting Pieces to the Maintenance Puzzle*. [S.l.]: IEEE Software, 1990. Citado na página 18.

PETERS, J. F.; PEDRYCZ, W. *Engenharia de Software: Teoria e Prática*. Rio de Janeiro: Campus, 2011. Citado 6 vezes nas páginas 6, 15, 18, 19, 20 e 21.

PFLEEGER, S. L. *Engenharia de Software: Teoria e Prática*. 2. ed. São Paulo: Prentice Hall, 2004. Citado 6 vezes nas páginas 10, 11, 15, 19, 20 e 21.

PRESSMAN, R. S. *Engenharia de Software: Uma Abordagem Profissional*. 7. ed. Porto Alegre (RS): AMGH, 2011. Citado 4 vezes nas páginas 15, 18, 20 e 21.

SILBERSCHATZ, A.; KORTH, H. F.; SUDARSHAN, S. *Sistema de banco de dados*. 6. ed. Rio de Janeiro: Elsevier, 2012. Citado 12 vezes nas páginas 15, 16, 22, 23, 24, 25, 26, 28, 29, 34, 46 e 55.

SOMMERVILLE, I. *Engenharia de Software*. 9. ed. São Paulo: Prentice Hall, 2011. Citado 6 vezes nas páginas 11, 15, 18, 19, 21 e 22.

UFU. *FAEPU: Fundação de Assistência, Estudo e Pesquisa de Uberlândia*. 2017. Disponível em: <http://www.faepu.org.br/>. Acesso em: 9 nov. 2017. Citado na página 43.

UFU. *HCU-UFU: Hospital de Clínica de Uberlândia - UFU*. 2017. Disponível em: <http://www.hc.ufu.br/>. Acesso em: 9 nov. 2017. Citado 2 vezes nas páginas 10 e 43.